

月刊マイコン別冊

Z80 マシン語 プログラム入門

PC-8001/PC-8801/mkII (N-BASICモード)

- マシン語の基礎知識
- Z80のインストラクション・セットI
- Z80のインストラクション・セットII
- マシン語による2次元4目ならべ
- N-BASIC 内部解析

FORESIGHT企画部 川村 清 著

マシン語の基礎からゲームプログラムまで

Hudson soft

ジャン狂

実力向上約束！
コンピュータ相手に君の雀力を発揮せよ

おもしろ

体験記



ジャン狂の作者
飛田雅宏

ゲームをやっていて何が面白いかって、そりゃあやっぱり偶然性でしょう。花札、オイチョカブ、はたまた馬券買い。なかでも麻雀の楽しさはやった者でないと分らない面白さなんだな。

麻雀の面白さがそっくりパソコンゲームになったのが今回発売の「ジャン狂」です。フルグラフィックでオールマシン語。しかも、相手となるコンピュー

タがなかなかの強敵。だから、挑戦者は熱くなってしまう。

そこで某月某日、たまたまハドソンに遊びにきた雀歴2年の君塚泰彦君(高3)にチャレンジしてもらいました。そばでニヤニヤ見学しているのは「ジャン狂」制作者の飛田雅宏氏(ハドソン・開発担当)。

ルールは2万7000点持ちの3万点返しという一般的ルール。最初に食タン有るかオープンモードかの選択がありますので好みの方をセレクトして下さい。オープンモードというのは相手の手をオープン状態

にしておくことで、麻雀研究に役立ちます。セレクトしたところでハイ、スタート。ところがこの段階で君塚君、最初の悲鳴！「ワッ、速いなー」マシン語ソフトの強味で、ツモまでの時間がいかにもスピーディー。次々に進行します。制作者の飛田氏。「このくらいのスピードだから面白いのです。でも、チョンボが出ない

よう、ナイたりロンの場合はコンピュータが一時停止機能を働せますからミスも少ない」と解説。そして、君塚君、フリ込んだりアガったりでゲーム終了の20分後にはマイナス1万3000点。「うーん、こいつは強いや」。ゲーム途中でエスケープキーを押した後にリターンキーを押すとオープンモードの切換えができますので麻雀の勉強には最高ですが麻雀は相手の顔つき、手つきな



どの様子を見るのも勝負のうちですが、コンピュータはまったくのポーカーフェイス。2、4、6、8とスペースキーだけの簡単操作です。あなたの雀力はどれくらいかな、チャレンジして下さい。



おもしろ体験者の
君塚泰宏君(高3)

適応機種

| | | | | | |
|------------------|------|--------|-------------|------|--------|
| PC-9801 | 8"FD | ¥7,800 | PC-6001 | T | ¥4,000 |
| PC-9801 | 5"FD | ¥6,800 | PC-6001mkII | T | ¥4,000 |
| PC-9801F | 5"FD | ¥6,800 | FM-7 | T | ¥4,000 |
| PC-8801/mkII(共用) | 5"FD | ¥6,800 | FM-7 | 5"FD | ¥6,800 |
| PC-8801/mkII(共用) | T | ¥4,000 | X1-D | 3"FD | ¥6,800 |
| PC-8001mkII | T(予) | ¥4,000 | X1-C | T | ¥4,000 |



ゴルフ狂

君はシングル?パーティーを
目指してティーショット!

おもしろ

体験記

Hudson Soft



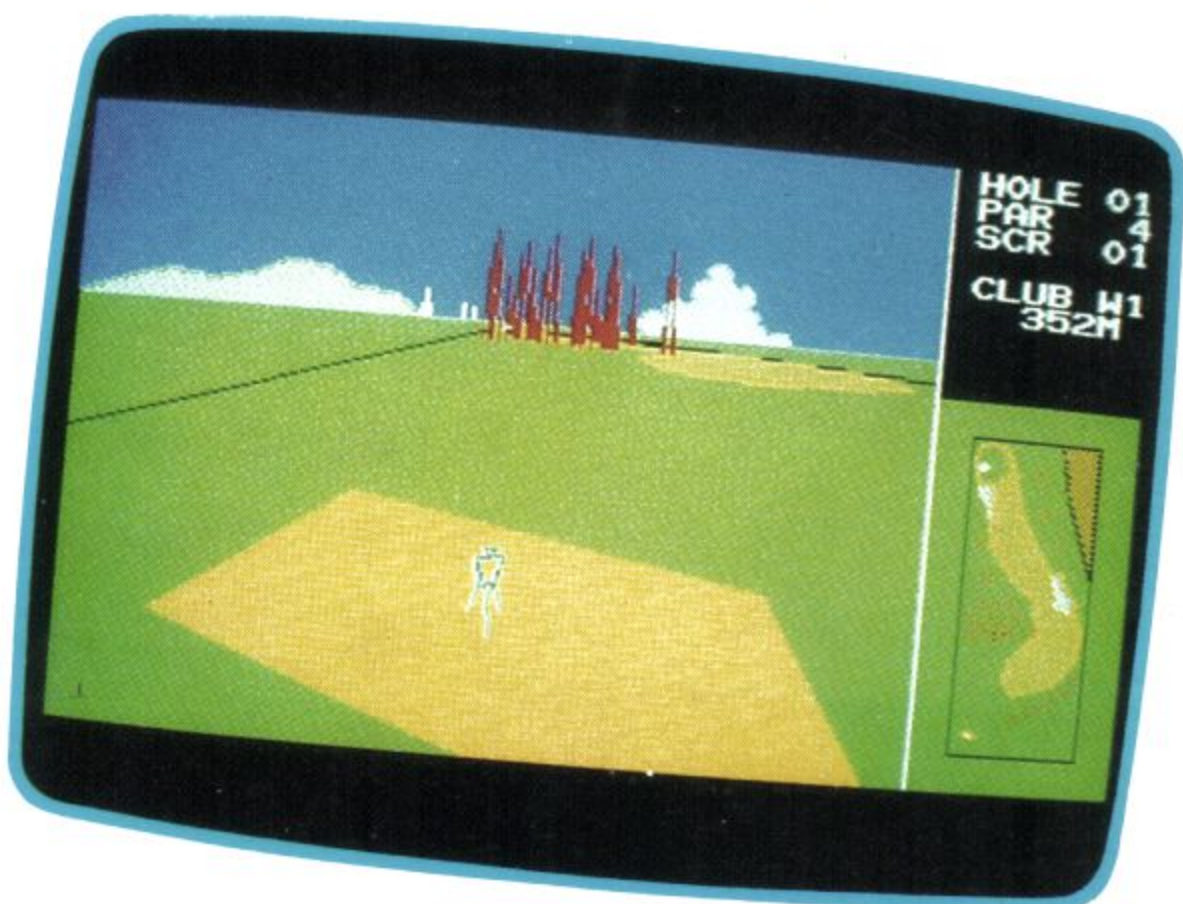
ゴルフ狂の作者 田中裕二

サアノここはハドソン・カントリー・クラブ。むこうに見えるはティネ山?只今から、本格的3次元ゴルフシミュレーションゲームの始まり始まり。プレーはアニメゴルファーのT選手。なあんて、実際に画面上でゴルフプレーが展開される。こんなゲーム、今まで見たことある? サスガコンパクトフロッピー版。

この本モノそっくりのゴルフゲームにチャレンジするは、パソコンに関してはちとうるさい小久保秀次郎君高一。

小久保君「画面の中、むこうにホールのピンが見える。ハイ、ゴルファーのステップ位置はこれでよし。この角度ならピッタリ。打球の強さはと……。ゴルファーが1回だけ素振りをするから、2回目のスイングのときにテイクバック

(ホラ、勢いをつけるために、クラブを振りかぶるアレだよ)のタイミングを見はからって、エイノと、スペースバーをたたくと、打球はアレ? O Bだ……。エ? ティーショットはアイアンで打つとフックがかかり易くなるんだって? 上空の風にあおられちゃっ



おもしろ体験者
小久保秀次郎君(高一)

たのか、ゴルフなんて初めてだからなあ。」

さて、小久保君、ウッドでやり直して、途中ラフに入れながらも、何とか池越えに成功。

ここでもう一度あたりの状況を見ると……。ワンタッチで画面は広角に。広い視野をカバーできるんだ。アノ右横にバンカーがあるぞ。そこで、左に寄せて……と、アレーノ左にもバンカー。

ハアノバンカーはサンドエッジで強く出すんだって? やっとグリーンにノったけど、ピンまで遠いなあ。いつになったらこのホール……。結局、パー4のところ、倍の8打たたいちゃった。でも確かにホントにゴルフをやってるような感じも。こりゃ慣れてきたらパーティーもホールインワンも夢じゃないぞ。」(私は全ホール回って7アンダーでした。作者)

感想「このやり方なら、マイコンボーリング、マイコンテニス、マイコン……。何だって楽しくなると思うなあノ」

適応機種

| | | | | | |
|------------------|------|--------|------|------|--------|
| PC-9801 | 8"FD | ¥7,800 | FM-7 | T | ¥4,000 |
| PC-9801 | 5"FD | ¥6,800 | FM-7 | 5"FD | ¥6,800 |
| PC-9801F | 5"FD | ¥6,800 | X1-D | 3"FD | ¥6,800 |
| PC-8801/mkII(共用) | 5"FD | ¥6,800 | X1-C | T | ¥4,000 |
| PC-8801/mkII(共用) | T | ¥4,000 | | | |

HUDSON GROUP

HUDSON SOFT®

ハドソン札幌/〒062 札幌市豊平区平岸3条5丁目4番17号 コロナード平岸II 201 PHONE:011-821-1538
ハドソン仙台/〒980 宮城県仙台市宮町1丁目4番28号 PHONE:0222-65-7031
ハドソン金沢/〒920 石川県金沢市本町2丁目1番28号 PHONE:0762-23-1263
ハドソン東京/〒102 東京都千代田区麹町4丁目7番5号 麹町ロイヤルビル2F PHONE:03-234-4996
ハドソン大阪/〒542 大阪市南区南船場4丁目2番18号 佐野屋橋ビル5F PHONE:06-251-1945
ハドソンUSA/2063 CENTER STREET BERKELEY CA 94704 TELEPHONE 415-845-1416

PC-8001・PC

Z80 マシン語プログラム入門

目次

第

1

ブロック

マシン語の基礎知識

| | | |
|-----|--------------|----|
| 第1章 | コーディングの実際 | 8 |
| 第2章 | アセンブルの実際 | 12 |
| 第3章 | Z80のレジスタ・セット | 18 |
| 第4章 | ミニ・ブレーク・ポインタ | 25 |

第

2

ブロック

Z80のインストラクション・セット I

| | | |
|------|---------------------|----|
| 第5章 | 8ビット・ロード命令 | 34 |
| 第6章 | 16ビット・ロード命令 I | 40 |
| 第7章 | 16ビット・ロード命令 II | 46 |
| 第8章 | 8ビット算術演算命令 | 52 |
| 第9章 | 8ビット論理演算命令 | 59 |
| 第10章 | インクリメント命令／ディクリメント命令 | 65 |
| 第11章 | 16ビット算術演算命令 | 70 |
| 第12章 | ジャンプ命令 | 75 |
| 第13章 | 相対ジャンプ命令 | 80 |
| 第14章 | 5本のプログラム例 | 84 |

第

3

ブロック

Z80のインストラクション・セット II

| | | |
|------|---------------------|----|
| 第15章 | サブルーチン・コール命令／リターン命令 | 88 |
|------|---------------------|----|

| | | |
|------|---------------------------------|-----|
| 第16章 | ローテート・シフト命令 I | 100 |
| 第17章 | 8ビット乗算プログラム | 104 |
| 第18章 | ローテート・シフト命令 II | 108 |
| 第19章 | EXAMINATION | 113 |
| 第20章 | CPUコントロール命令 | 118 |
| 第21章 | 入出力命令 | 122 |
| 第22章 | マシン語 Q & A | 129 |
| 第23章 | ブロック転送命令 / ブロック・サーチ命令 / ビット操作命令 | 135 |
| 第24章 | サーチ・メモリ・プログラム | 140 |
| 第25章 | マイクロ・ワード・プロセッサ I | 148 |
| 第26章 | マイクロ・ワード・プロセッサ II | 151 |

第

4
ブロック

マシン語による2次元4目ならべ

| | | |
|------|--------------------------|-----|
| 第27章 | 2次元4目ならべ | 160 |
| 第28章 | 汎用サブルーチン・パッケージ・アセンブル・リスト | 177 |
| 第29章 | サブルーチン・アセンブル・リスト | 188 |
| 第30章 | メインルーチン・アセンブル・リスト | 210 |

第

5
ブロック

付録

| | | |
|-----|---------------------------|-----|
| 付録A | μCOM-82インストラクション活用表 | 238 |
| 付録B | N-BASIC 中間言語 & 処理アドレス一覧表 | 254 |
| 付録C | N-BASIC 内蔵モニタ処理アドレス一覧表 | 256 |
| 付録D | N-BASIC システム・サブルーチン一覧表 I | 257 |
| 付録E | N-BASIC システム・サブルーチン一覧表 II | 260 |
| 付録F | N-BASIC システム・ワークエリア一覧表 | 262 |
| 付録G | μPD3301アトリビュート・コード表 | 266 |
| 付録H | キャラクタ・コード表 | 267 |
| 付録I | 参考文献 & 引用文献一覧表 | 267 |

マイコン別冊

GAMINGへの招待

あ・な・た・も

テレビゲーム・プログラムに挑戦

PC-8001/8801(N-BASICモード)

好評発売中

あなたもテレビゲーム・プログラムに挑戦

GAMING

GAMINGへの招待

MULTIマイコン研究会 塚越 一雄 著

SPACE WAR



1. SPACE WAR

2. テニスゲーム

3. BOMBER

4. インバーダー・パニック

ゲームプログラム
ホイ
ホイ



「GAMINGへの招待」で
君も今日から
ゲームプログラマー!

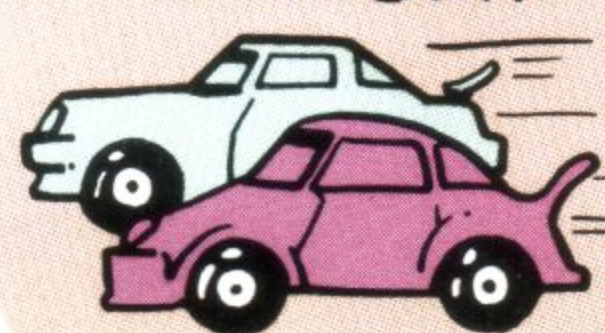


スペースウォー、テニス、ボンバー、インバーダー
4ジャンルのゲームプログラムのノウハウを全公開
BASICからマシン語までゲームプログラム入門!

テニス ゲーム



BOMBER



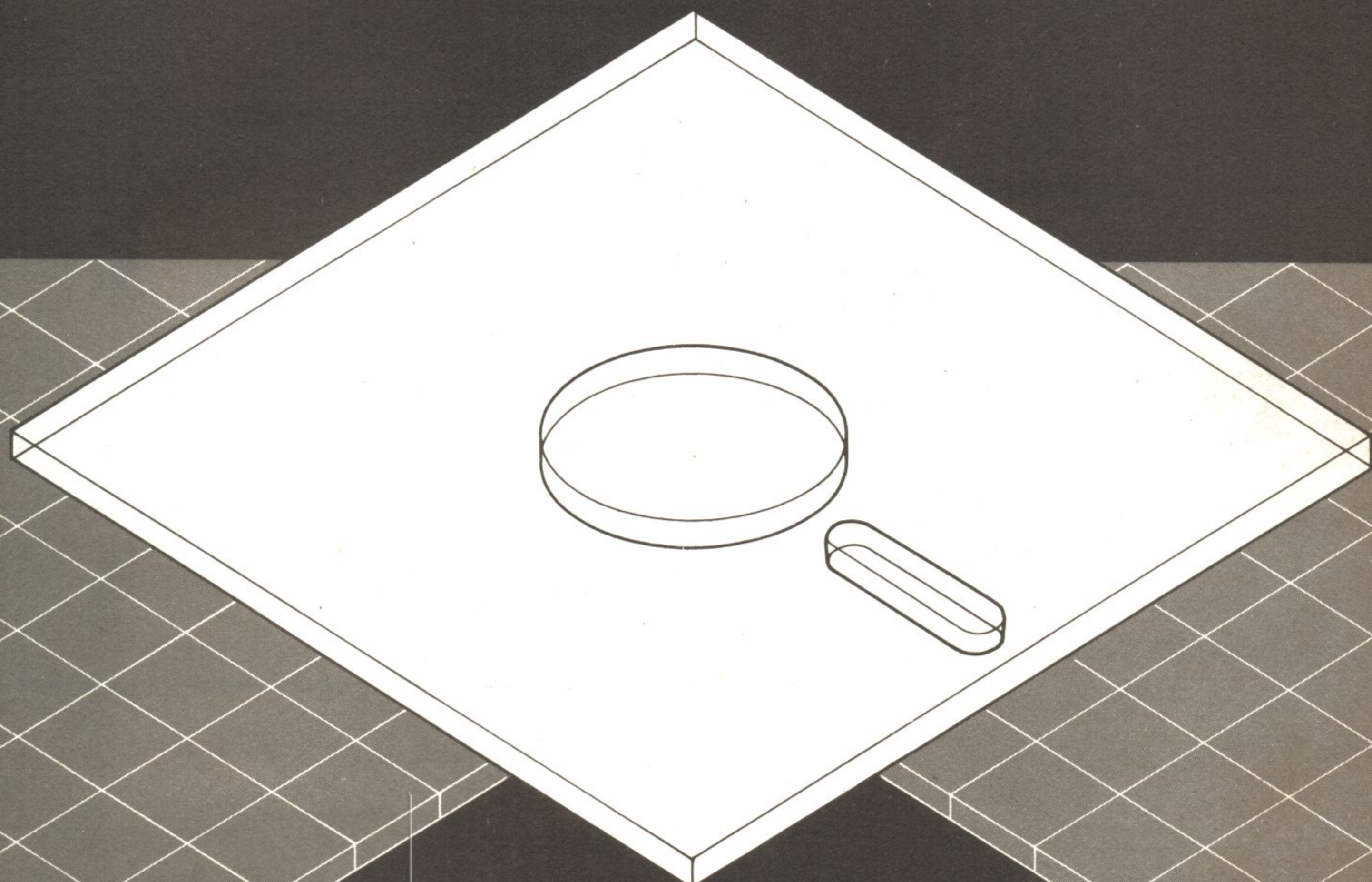
B5判 220頁 定価1,300円 (送料250円)

電波新聞社 出版販売部

御注文は各本社、支局またはマイコンショップ・書店で。

東京本社 〒141 東京都品川区東五反田1-11-15 (03-445-6111)
大阪本社 〒530 大阪市北区中之島3-2-4 (06-203-3361)
西部本社 〒812 福岡市博多区博多駅前2-13-23 (092-431-7411)

第1 ブロック



マシン語の基礎知識

コーディングの実際

私は今日まで、コンピュータに関する資料や文献をあれこれ数限りない程開いてきましたが、大部分の文献には勉強させられる事が多く大変ためになります。

しかし、最近はともかくとして、私がコンピュータを学ぼうと志したころの状況を思い出すと、どのような文献をみてもなかなか取っ付きにくく、思考錯誤の連続であったことを記憶しています。

一般に、専門分野における入門書が入門者によって執筆される事は皆無だと思えます。ほとんどの入門書は各専門分野の第一戦で長い間活躍してきた人達や、研究・教育にたずさわってきた人達によってつくられているのが現状です。

よく、他人を教育するには初心者立場に立って教える事が必要だと言われますが、私が入門書を執筆する事になったからといって、今さら初心者にもどる事は不可能でしょう。

確かに、おもしろおかしい文章表現を用いて読者の気を引くことも、一つの手段であると思います。しかし本書ではでき得る限りきめの細かい解説を行う事によって文章のつたなさをカバーして行きたいと考えています。

画面設定

当分の間、画面操作に関する説明は、全て白黒モード、80字(80桁)×25行モードを想定

して行います。

つまり、PCのスイッチを入れたあと、

```
CONSOLE 0, 25, 0, 0CR
```

```
WIDTH 80, 25CR
```

を行えばOKです。

^{C_R}は、RETURNキーです。

前置きが長くなりましたが、そろそろ本題に入りましょう。

PC-8001のテレビ画面

第1図を見てください。これはPC-8001のテレビ画面、左上の隅 LOCATE 0, 0の位置にキャラクタ・グラフィックの“●”を表示するためのプログラムを、POKE文を用い、BASICで組んだものです。

PCには、増設分も含めて、8000H番地から、FFFFH番地までの32Kバイト分、DRAM(ダイナミック・ラム)が実装できますが、その内のF300H番地から、FEB7H番地までの3Kバイトは、V-RAM(ビデオ・ラム)

《第1図》BASIC

```
10 LET A=&HEC
20 POKE &HF300, A
30 END
```

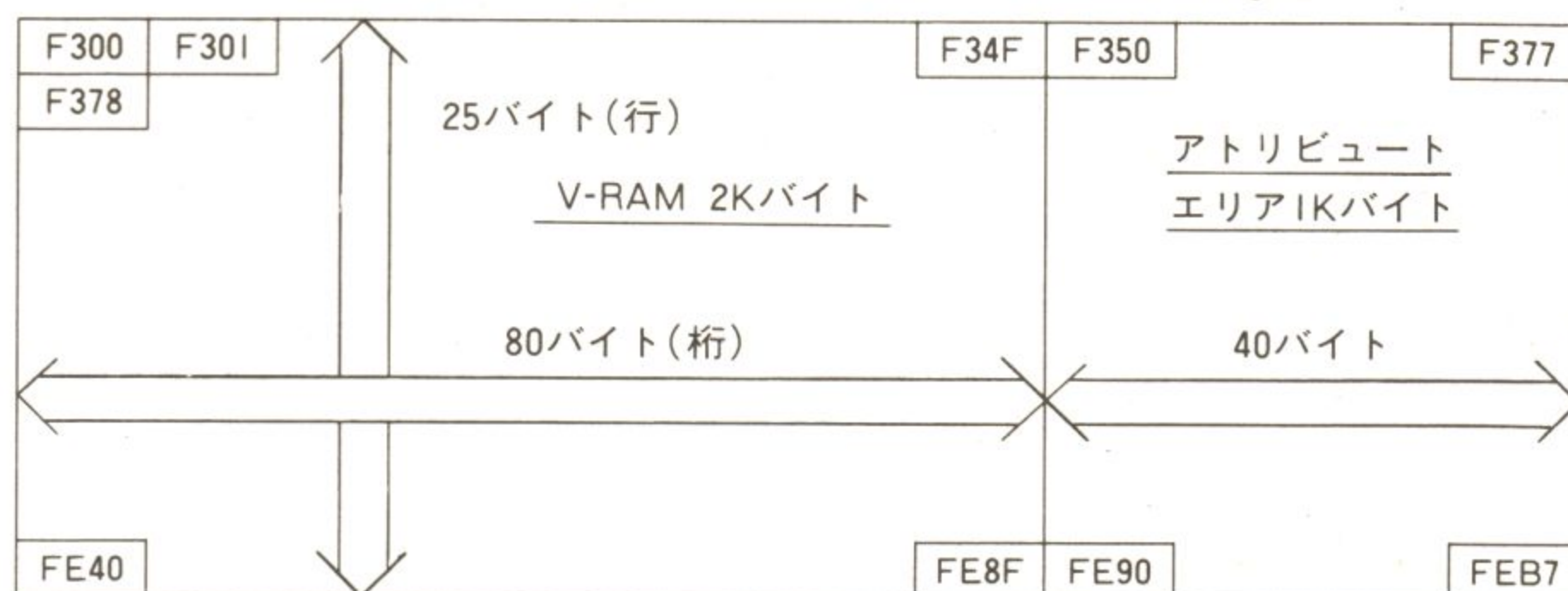

になっています。

第2図に、V-RAMとテレビ画面の対応を示します。

この表でもわかるように、V-RAMの1バイトが、テレビ画面の1文字に対応していますが、PCの画面が、80文字／行なのに対して、V-RAMの方は1行が120バイトになっています。

これは、PCがアトリビュート方式を採用しているためで、120バイトの内、前80バイトを除いた残りの40バイトはアトリビュート・エリアと呼ばれ、キャラクタ、グラフィック、カラー、アンダーライン、オーバーライン、反転、点滅……etc, などのコントロールに使用され、PCの特色あるオペレーションを可能にしています。

《第2図》PCのV-RAMとテレビ画面の対応



さて、アトリビュート・エリアを除いた80バイト×25行の2Kバイトですが、この部分が純粋なV-RAMと言えます。

ここで、V-RAMとテレビ画面の関係をテス

トしてみましょう。

まず、"MON^C_R"でマシン語のモニタに制御移してください。

次に、"SF300^C_R"を行ったあと、適当なキ

ャラクタ・コードを、16進数2桁で入力してってください。キャラクタ・コードと、表示されるキャラクタの対応は、第3図を見てくればわかると思います。

例えば"●"のキャラクタ・コードは、16進でEC,"K"のキャラクタ・コードなら4Bです。

このテストでもわかるように、V-RAM上にキャラクタ・コードを書き込むと、それに対応したキャラクタがそのままTV画面上に表示されるのです。

V-RAM上にコードを書き込むには、

- ①、モニタの、S(セット)コマンドを使う
- ②、BASICのPOKE文を使う
- ③、マシン語を使う

など、幾つかの方法がありますが、①は、先のテストの時使い、②は第1図で使っています。

《第3図》PC-8001のキャラクタ・コード

上位4ビット→

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|----|----|----|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | DE | | 0 | @ | P | | p | | ┌ | | 一 | タ | ミ | ≡ | × |
| 1 | SH | D1 | ! | 1 | A | Q | a | q | | └ | | 。 | ア | チ | ム | 円 |
| 2 | SX | D2 | !! | 2 | B | R | b | r | | ┌ | | 「 | イ | ツ | メ | 年 |
| 3 | EX | D3 | # | 3 | C | S | c | s | | └ | | 」 | ウ | テ | モ | 月 |
| 4 | ET | D4 | \$ | 4 | D | T | d | t | | ┌ | | 、 | エ | ト | ヤ | 日 |
| 5 | EQ | NK | % | 5 | E | U | e | u | | └ | | ・ | オ | ナ | ユ | 時 |
| 6 | AK | SN | & | 6 | F | V | f | v | | ┌ | | ヲ | カ | ニ | ヨ | 分 |
| 7 | BL | EB | ▼ | 7 | G | W | g | w | | └ | | ア | キ | ヌ | ラ | 秒 |
| 8 | BS | CN | (| 8 | H | X | h | x | | ┌ | | イ | ク | ネ | リ | ♠ |
| 9 | HT | EM |) | 9 | I | Y | i | y | | └ | | ウ | ケ | ノ | ル | ♥ |
| A | LF | SB | * | : | J | Z | j | z | | ┌ | | エ | コ | ハ | レ | ♦ |
| B | HM | EC | + | ; | K | [| k | { | | └ | | オ | サ | ヒ | ロ | ♣ |
| C | CL | → | , | < | L | ¥ | l | : | | ┌ | | ヤ | シ | フ | ワ | ● |
| D | CR | ← | = | = | M |] | m | } | | └ | | ユ | ス | ヘ | ン | ○ |
| E | SO | ↑ | . | > | N | ^ | n | ~ | | ┌ | | ヨ | セ | ホ | 〃 | ◀ |
| F | SI | ↓ | / | ? | O | — | o | | | └ | | ツ | ソ | マ | ° | ▶ |

下位4ビット→

本章では、③の方法を説明したいと思います。

逆に、V-RAMの内容を調べれば、現在、テレビ画面にどんなキャラクタが表示されているかわかるわけで、V-RAMの内容を読み出すには、

- ①、モニタのD (ダンプ) 又は、S (セット) コマンドを使う
- ②、BASICのPEEK文を使う
- ③、マシン語を使う

などの方法があります。

第1図の説明

第1図について説明します。

10行では、LET文を使って変数Aに16進数のECを代入しています。ECは“●”のキャラクタ・コードです。また、“LET”は省略することが多いのですが、ここでは第4図と比較し易いような形にしてあります。

20行は、先程の②の方法を使って、変数Aの内容(ECH)を、F300H番地に代入しています。

F300H番地は、V-RAMの先頭番地であり、テレビ画面上では、左上隅、LOCATE 0, 0の位置です。

30行の“END”も省略できます。

このプログラムをRUNさせると、画面の左上隅に“●”が表示されるはずですが、PRINT文と違って、アトリビュート・エリアまで書き換えるのではないので、その位置のカラー指定が、シークレットや黒になっている時には、当然表示されないので注意が必要です。

第4図の説明

ここからが本当の、マシン語講座の内容です。

第1図と第4図を比べて見てください。

前者と同じ内容のプログラムをニーモニック(アセンブリ言語)で組んだものだが、後者のプログラムです。

「マシン語でプログラムをつくる」とはとっても、直接16進数のら列を書いたり理解したりするのはたいへんですから、直接マシン語で組んでいく必要はありません。

まず一度、ニーモニックを使って、プログラムを組んでいき、ニーモニック↔マシン語の対応表を引いていくなり、アセンブラを使用するなりして、マシン語に直すのが普通です。

ここで、ニーモニックからマシン語に直すのをアセンブルといい、完成したマシン語をオブジェクトといいます。

逆にマシン語からニーモニックを作り出すのを逆アセンブル(ディス・アセンブル)といい、それをプログラムで行ってしまおうというのを逆アセンブラ(ディス・アセンブラ)といいます。

第1図と第4図を比較していくと、両者の形成がそれ程かけ離れていない、どちらかというと同じような形をしていることに気がつくと思います。

読者の皆さんの中には、マシン語は難しいと思ひ込み、マシン語コンプレックスに落ち込みかけている方もいるかもしれませんが、この程度のプログラムなら、BASICで組んでもマシン語で組んでも、大きな差はみられません。

マシン語は、難しいというよりも、少し手間がかかるだけなのです。

では、第4図を行を追って説明していきます。

一番最初にある“LD A, ECH”のLDというのは、ロードと読み BASICのLET文と同じく代入文です。この例では、“A”に16進数のECを代入しています。

この“A”はBASICの変数と似ていますが、マシン語ではレジスタ(Reg)の一つで、Aレジスタとか、アキュムレータ(Acc)とか呼ばれているもので、8ビット(16進2桁)です。

《第4図》ニーモニック

```
LD    A, ECH
LD    (F300H), A
HALT
```

PC-8001が使用しているZ80CPUは、合計22個のレジスタ群を持っていますが、次章以

降で少しずつ説明していく予定です。

ECHの最後のHは、ECが16進数であるということで、BASICで16進数の頭に付ける“&H”と同じ意味です。全くの余談ですが、6800CPUなどは頭に“\$”を付けて16進数を表わします。

第4図の2行目の“LD (F300H), A”もやはりLD命令が使われていますが、F300Hに(カッコ)が付いています。Z80のニーモニック中で、16進4桁の数値に(カッコ)が付いている場合には、その数値が、メモリ上のアドレスを指していると考えればよいのです。

ですからこの命令は、BASICのPOKE文と同じように、F300H番地にAレジスタの内容を入れるという働きをします。

最後に“HALT”という命令が置いてあります

がこれは、Z80CPUを停止する命令です。ですからプログラムの最後にこの命令をいれておけば、Z80は、そこまでプログラムの実行を停止します。

停止を解除できるのは、リセットスイッチのみです。

まとめ

第1章は、ここまでですが、このニーモニックを直接PC-8001に入力して走らせることはできません。

先程の“アセンブル”という作業が残っていますが、長くなるのでこれについては第2章で説明したいと思います。

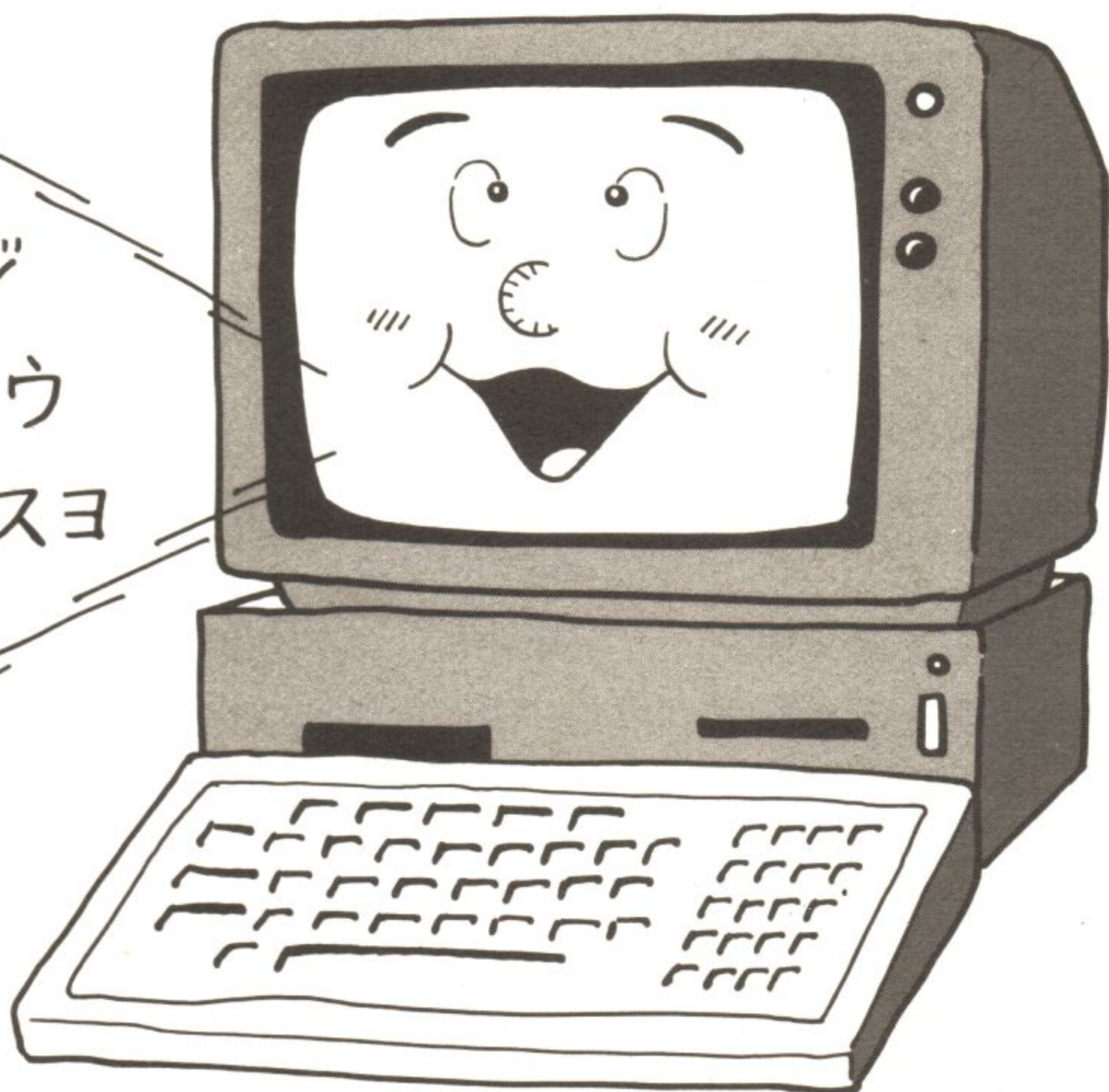
```
10 LET A=&HEC
20 POKE &HF300,A
30 END
```

BASIC言語

```
LD A,ECH
LD (F300H),A
HALT
```

アセンブリ言語

オナジ
ナイヨウ
デスヨ



アセンブルの実際

第1章、いかがだったでしょうか？

前章では、

“LD □, △”

という命令は、△の内容を□に代入するということを説明しました。またこの時4桁の16進数に()が付いていれば、それはアドレスを指していて、

“A”と書けば、それはAレジスタ（アキュムレータ）のことでしたね。

PC-8001のメモリ・マップ

アセンブルを行う時には、まず完成したマシン語を、メモリ上のどの番地に置くかを決めなければなりません。

第5図にPC-8001のメモリ・マップを示します。左が16KバイトRAMのシステム、右がRAMを32Kバイトに増設したシステムですが、今回は16Kバイトのシステムを例にとって説明します。

0000H～7FFFFHは、PC-8001の場合はROMのための領域で、この内、

0000H～5FFFFHの24Kバイトは、BASICのインタプリタが入っており残りの8Kバイトは、増設用の空きソケットになっています。

以上ROM領域には、簡単にプログラムを書き込むことはできず、ROMライターを使うかハードウェアをいじらなければなりません。

8000H～FFFFHは、RAM領域になっ

ていますが、16Kシステムの場合は、C000H～FFFFHの16KバイトのみRAMが実装されていて8000H～BFFFFHは空いています。

RAM領域の内、F300H～FEB7Hの3Kバイトは、前章でも説明したV-RAM領域で当然、プログラム・エリアとしては使えません。

オール・マシン語のみのプログラムを作り、BASICにももどらず BASIC インタプリタ内のサブルーチン類なども絶対使わない自信があるなら、V-RAMだけを避ければよいのですが、ほとんどの場合そうではないので、EA00H～FFFFHのN-BASICのワーク・エリア（V-RAMも含む）と、RAMの頭から数10バイトは、使わない方が無難です。

使おうと思えば空いているところもあるのですが、下手に使うと暴走させることにもなりかねません。

以上長々と説明して来ましたが、結局、16Kバイトシステムの場合は、C020H～E9FFH、32Kシステムでは、8020H～E9FFHの範囲で使えばよいでしょう。ディスクを使っている場合は、更に使用範囲が限られてしまいます。

使用可能領域を第5図中にアミ部分で示しますので参考にしてください。

N-BASICの CLEARコマンド

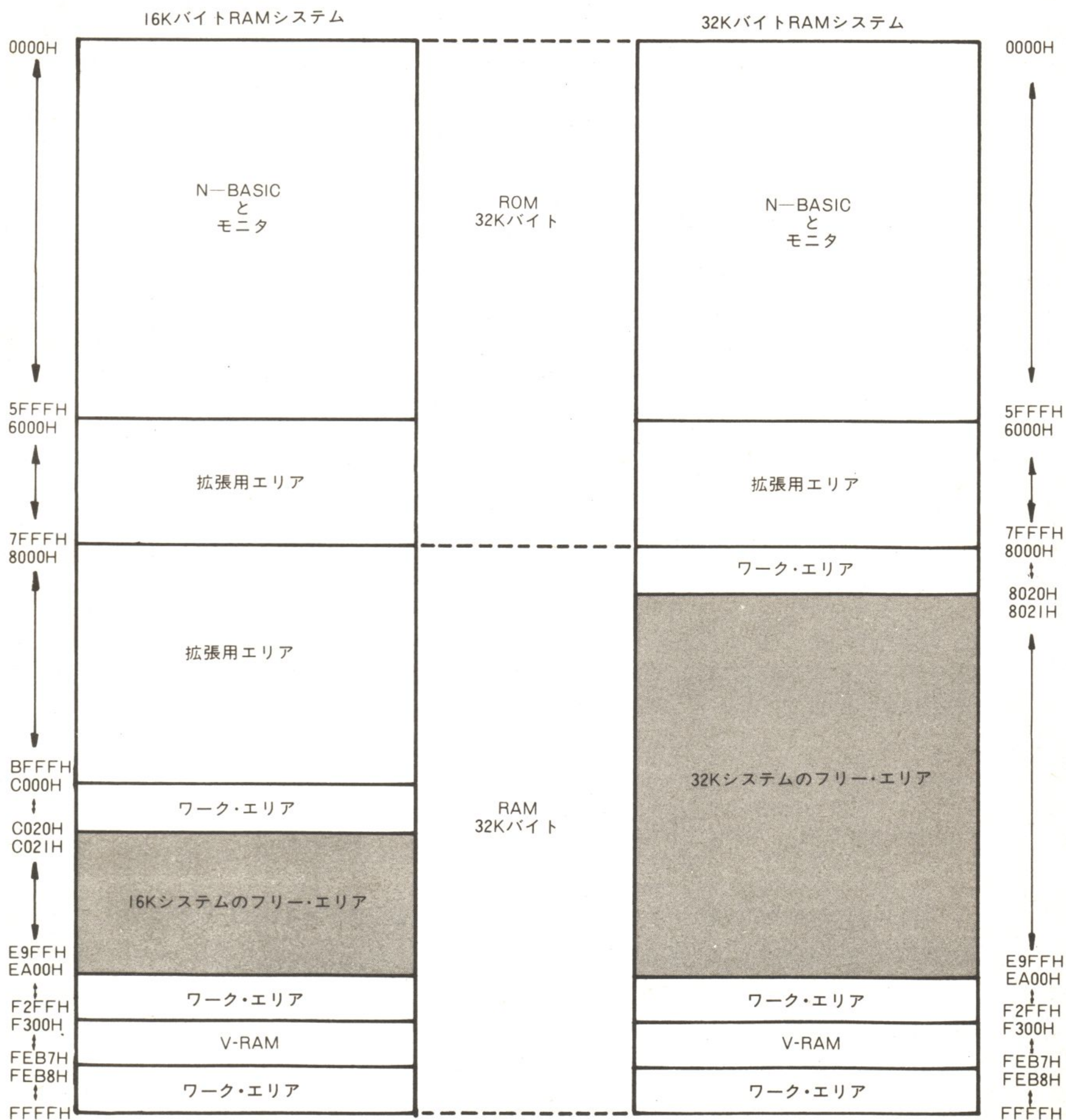
マシン語のプログラム領域を確保するコマンドとして、N-BASICのCLEARコマンドがあります。このCLEARコマンドで、マシン語のために使う領域を確保して置けば安全ですが、それでもフリー・エリア以外を使うことはできません。

例えば、ダイレクト・モードで、

CLEAR 300, &HFFFF^{C_R}

を実行して置けば、E000H番地から、E9FFH番地までをマシン語のために確保したことになり、N-BASICではRAMの先頭から、DFFFH番地までと、EA00H番地からFFFFH番地のみを使用します。

《第5図》PC8001メモリ・マップ



アセンブル

それでは、第6図をE000H番地からのマシン語にアセンブルしてみましょう。このプログラムは、第1章の第4図と全く同じもので、テレビ画面、左上の隅に“●”を表示するプログラムです。

アセンブラを持っている場合には、それを使えば簡単にアセンブルできますが、そうでない場合には、ニーモニック→マシン語の対応表を引きながら、自分でアセンブルしなければなりません。

《第6図》

| | |
|------|------------|
| LD | A, ECH |
| LD | (F300H), A |
| HALT | |

この対応表は、本書も含む文献の付録などという形でもよく見かけますが、このようなものとは別に表だけのものを持っていると大へん引き易く、安いものですから一冊買った方がよいと思います。

ちなみに私は、NECで出ている“μCOM-82インストラクション活用表”というのを使っているのですがこの対応表に合わせて、説明を行います。μCOM-82というのは、Z80のことです。

《第8図》第6図をアセンブルしたもの

| アドレス (番地) | オブジェクト (マシン語) | ラベル | ニーモニック | | コメント(メモ) |
|--------------|------------------|-----|--------|------------|------------------------|
| | | | コード | オペランド | |
| E000 | 3EEC | | LD | A, ECH | アキュムレータにECHを入れる |
| E002 | 3200F3 | | LD | (F300H), A | F300番地にアキュムレータの内容を代入する |
| E005 | 76 | | HALT | | 実行を停止する |

《第7図》μCOM-82ニーモニック↔マシン語対照表(8ビットロード)より

| | × | I | R | A | B | C | D | E | H | L | (HL) | (BC) | (DE) | (IX+d) | (IY+d) | (n n) | n |
|-------------|---|-----------|-----------|--------------|-----|-----|-----|-----|-----|-----|------|------|------|----------------|----------------|---------------|----------|
| | | | | | | | | | | | | | | | | | |
| LD A, × | × | ED 5 7 | ED 5 F | 7 F | 7 8 | 7 9 | 7 A | 7 B | 7 C | 7 D | 7 E | 0 A | 1 A | DD 7 E d | FD 7 E d | 3 A n n | 3 E n |
| LD (n n), × | × | | | 32 n n | | | | | | | | | | | | | |

1行目の“LD A, ECH”を調べるには、“ECH”を“n”に変えて、8ビット・ロード命令の項を見ます(第7図)。

対応表などでは、1バイト(16進数で2桁)定数は、全て“n”などのように置き変えて載っているはずですが、2バイト(16進数で4桁)の場合は、“nn”などになります。

ですから、“LD A, ECH”は、“LD A, n”という形に直して調べ、後で“n”の部分に“EC”を、入れてやればよいのです。

その結果“3E n”のように出ているはずですが、この“n”の部分に“EC”に直して、“3E EC”となります。ですから“LD A, ECH”のマシン語は、“3E EC”の2バイトで、これをそれぞれ、E000H番地とE001H番地に割り当てます。

次の“LD (F300H), A”も同じく、“LD (n n), A”と直して、対応表の8ビット・ロード命令の項で見ます。

結果は、“32 n n”ですがここで、

“32 F3 00”としては、いけません!

Z80や8080などのCPUでは、F3と00を逆にして、“32 00 F3”のようにしなければいけないので注意が必要です。特にはじめのうちは、間違え易いので気を付けてください。

この3バイトを、E002H~E004H番地に割り当てます。

最後の“HALT”は、マシン語では、“76”ですからE005H番地は、76になります。

いよいよ実行！！！！

以上のニーモニックとマシン語をきちんと表にしたのが、第8図ですが、この表について説明する前にプログラムを実行してみましょう。

BASICから“MON^C_R”で、マシン語モニタに移してください。“SE000^C_R”を行った後、“3EEC3200F376”と、6バイトのプログラムを入力します。もし間違えて入力した時には、“DEL”キーを押すことによって1バイトもどって入力し直すことができます。

いよいよ実行です。“GE000^C_R”でE000H番地から実行させてみます。画面左上に“●”が表示されればOK、そうでない方は、入力ミスだと思いますので、リセット・スイッチを押して始めから入力し直してください。

マシン語のコーディング書式

マシン語をコーディングする時には、第8図のような書き方をするのが普通です。マイコン誌上でもよくこのような形のリストが掲載されているのでご存知の方も多いと思いますが一応説明します。

まず一番左にアドレスを書きます。これは、その名のとおりメモリ上の番地で、マシン語をコーディングする時は普通1行に一つの命令を書きますから、その命令の始まる番地を16進数で書きます。

次に普通は、マシン語を書きますが、ハンド・アセンブルなどをする場合で、縦に罫を引いていないノートなどにコーディングする場合には、マシン語を1番右に書くこともあります。

いずれにしても、第8図のような書き方のみに決められているわけではなく、要は、必要な事項がわかり易く書かれていればよいのです。第10図に市販されているコーディング用紙の中で一般的なものをあげておきます。

マシン語の項は、今回のプログラムでは、3バイト分ではよいのですが、Z80には、1～4バイトの命令があるので、最大4バイト分の場所を取っ

《第9図》第6図をサブルーチンの形にしてアセンブルしたもの

| アドレス | オブジェクト | ハーモニック | コメント |
|------|----------|---------------|-------------------------|
| E000 | 3E EC | LD A, EC | アキュムレータにECHを入れる |
| E002 | 32 00 F3 | LD (F300H), A | F300H番地にアキュムレータの内容を代入する |
| E005 | C9 | RET | メインルーチンへもどる |

《第10図》一般的なコーディングの例

①市販品のコーディング用紙

| M・CODE | | MNEMONIC・CODE | | | |
|--------|-------|---------------|----|---------|---------------|
| LOC | INS | LABEL | OP | OPERAND | COMMENT |
| E000 | 3E EC | | LD | A, ECH | AレジスタにECHを入れる |

②ノートなどを使うとき 1

| アドレス | ラベル | ニーモニック | オペランド | コメント | オブジェクト |
|------|-----|--------|--------|-------|--------|
| E000 | | LD | A, ECH | A←ECH | 3E EC |

③ノートなどを使うとき 2

| | | | |
|------|----------|--------------|-------|
| E000 | LABEL:LD | A, ECH;A←ECH | 3E EC |
|------|----------|--------------|-------|

ておけば大丈夫です。

次にラベルと書いた欄がありますが、これは、ジャンプ命令の飛び先などにわかり易い名前を付けておくものです。このラベルについては、後でまた出てきますから、今は、それほど気にしないで結構です。

次のニーモニックと書いた欄には、先程の第6図がそのままの形で入っています。

ニーモニックの先頭の命令の種類を表わす部分をコード、以後のその命令が、どこに影響するか表わす部分をオペランドと言い、“,”の前を第1オペランド、“,”の後を第2オペランドと言います。

Z80には、オペランドが0～2個の命令があります。

その他、コードの部分をニーモニックと言っている場合もあります。

最後のコメントは、その名のとおりコメントを書く所で、プログラムがわかり易いように好きなことを書けばよいし、省略してもよいのです。

コーディング上の注意

マシン語をコーディングする上で注意しなければならないこともいくつかあります。

まず、字体ですが、例えば、Zと2、Oと0とDなどを区別して書くことが重要です。マシン語に限らず他の言語でコーディングする時でもできるだけ**第11図**のように書くくせをつけておくと便利です。ただし、学校のテストなどをこの字体でコーディング?してしまった時の責任を負いかねます(前例もあります)。

《第11図》コーディング用の字体

| | | |
|--------|-------|---------------|
| C | (シー) | カギを付ける |
| Ð | (ディー) | タスキをかける |
| i or Ī | (アイ) | 小文字を使うか上に線を引く |
| J | (ジェー) | 横線を引く |
| Ō | (オー) | 上に線を引く |
| u | (ユー) | 小文字を使う |
| Z | (ゼット) | タスキをかける |
| Ø | (ゼロ) | 斜線を入れる(当然!) |

また16進数を書くとき、そのバイト数を意識して1バイトか2バイトかわかるように2桁又は4桁で書くことを勧めます。

“2 H”という1桁の数値でも“LD A, 0 2 H”とか、“LD (0 0 0 2 H), A”とかいうように、前に余分な0を付けておくと、ハンドアセンブルの時のミス防止に役立ちます。

私も前に、“LD HL, 2”を間違えて、“2 1 0 2”の2バイトにアセンブルしてしまったことがあります。正しくは、“2 1 0 2 0 0”の3バイト命令です。このミスも“LD HL, 0 0 0 2 H”と書いておけば未然に防げたはずです。

アセンブラを使えばこのようなミスも起らないのですが、せっかくハンド・アセンブルした長いプログラムの真中などでこのミスをやると、後で1バイトずつずらしていかなければならないので大へんな手間になります。

Q&Aコーナー

このコーナーでは、初心者がマシン語を勉強していくうえでわかりにくい点を、1～2項目ずつ選んで説明していきます。私がマシン語を始めた頃の経験も生かしていきたいと思います。

◎BASIC中からのマシン語の呼び方

マシン語のプログラムをBASICのプログラム中から呼び出す方法を説明しますが、さっきマシン語に直したばかりの**第6図**を例にとります。

第6図をサブルーチンの形にしてアセンブルしたものが**第9図**です。先程のプログラムとの相違点は最後のHALT命令がRET命令にかわっているだけです。

HALTは、Z80 CPUを停止させる命令ですが、RETは、RETURNの略で、サブルーチンからメインルーチンへもどる命令です。

この場合は、マシン語のプログラム全体を一つのサブルーチンと考え、BASICのメインルーチンから呼び出すのです。



《第12図》マシン語実行用のBASICルーチン

```

10  CLEAR 300, &HFFFF
20  DEF USR=&HE000.
30  I=USR(0)
40  END

```

このプログラムを呼び出すための BASIC のメインルーチンが第12図です。

まず10行の CLEAR コマンドでマシン語のための領域を確保します。

次に DEF USR コマンドでマシン語プログラムのスタート・アドレスを定義し、30行でマシン語プログラムを実行します。

30行中の変数 I と (カッコ) の中の 0 は BASIC とマシン語との引数ですが、今回のように引数が必要ない場合にも、一応何かを与えないとエラーになります。引数の受け渡しは、浮動小数点アキュムレータ (F0A4H~F0ABH) を介して行われますが、詳しい事は、“N-BASIC リファレンス・マニュアル” をご覧ください。

また、N-BASIC では、USR 0~USR 9 の計 10 個のマシン語サブルーチンを使うことができますが、今回のように番号を省略した場合は、USR 0 と解釈されます。

第9図のオブジェクトを E000H~E005H 番地にセットし、BASIC にもどしてから第12図を入力してください。

“RUN^C” で画面左上に “●” を表示した後 BASIC のコマンド待ちになります。

ついでに、第13図をあげておきます。これはマシン語をメモリ上に書き込む段階も BASIC で行ってしまうというものです。

このプログラムなら、BASIC の CSAVE, CLOAD コマンド 1 回のみでロード、セーブができますし、マシン語モニタに移してプログラムを入力する必要がありません。

```

-----
DEF USR = &HE000
-----
-----

```

```

-----
I = USR(0)
-----

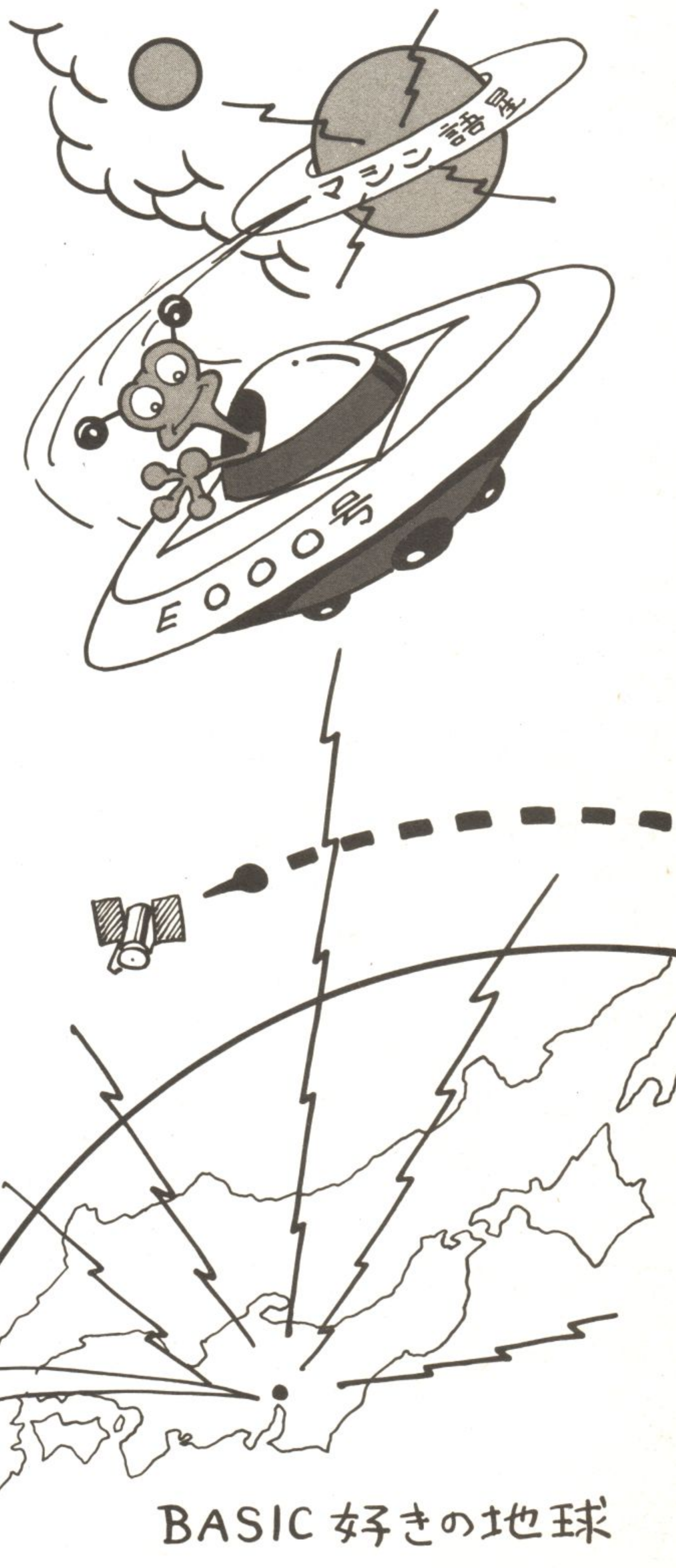
```

《第13図》BASICからマシン語を呼び出す

```

10  CLEAR 300 &HFFFF
20  DEF USR=&HE000
30  FOR J=&HE000 TO &HE005
40      READ K$
50      POKE J, VAL("&H"+K$)
60  NEXT J
70  DATA 3E, EC, 32, 00, F3, C9
80  I=USR(0)
90  END

```



Z80のレジスタ・セット

前章までのプログラム例ではAレジスタのみを使用するようにして来ましたが本章からはいよいよZ80のレジスタ群が登場します。

本章はこの22個のレジスタ達の自己紹介ということで一度に全てのレジスタが登場しますが惑わされないように注意してください。

あくまでも重要なのは「レジスタのまとめ」の項に書いてあることです。他のレジスタの説明は読み飛ばしても、「まとめ」に書いてある事だけは覚えておいてください。

Z80の汎用レジスタ

皆さんも前章までに数多く登場したAレジスタについては多少理解していただけたと思います。

Aレジスタはアキュムレータとも呼ばれる1バイトのレジスタで、1バイトの数値をBASICの変数のように記憶させておくことができます。

このようにレジスタというのは、いずれも数値を記憶させておく（しておく）ためにあるのですが、22個のレジスタ群全てが、Aレジスタのように、自由に使えるわけではありません。

第14-A図にZ80の全てのレジスタを示します。この内、Aレジスタを含めた、A、F、B、C、D、E、H、Lの8個のレジスタを表レジスタ群（メイン・レジスタ・セット）と呼んでいます。

このAレジスタを始めとする8個の8ビット(1

バイト)のレジスタが、マシン語でプログラムを組んで行く上での中心となるものです。

まずフラグ・レジスタ(F)ですが、このレジスタは、Aレジスタなどのように自由に数値を出し入れすることはできません。CPUが特定の演算などの命令を行うとその結果に従って勝手にフラグ・レジスタの内容が変更されてしまい、そのあとの条件ジャンプや条件コールなどの条件判断を行う時にこのフラグ・レジスタの内容が参照されるのです。

フラグ・レジスタを除いた7個の汎用レジスタの機能は、ほとんど同じですが、それぞれに得意な分野、不得意な分野があり、機能の点ではアキュムレータに勝るレジスタはありません。

例えば8ビットの算術・論理演算のほとんどは、アキュムレータと他のレジスタの間で行われ、その演算結果もアキュムレータに収納されるようになっているものがほとんどです。

残りのB、C、D、E、H、Lの6個のレジスタですが、BとC、DとE、HとLの2個ずつを対にして2バイト(16ビット)の、汎用レジスタとして使う命令も多く用意されています。

実際のプログラムを考えてみると、255(1バイト)以下の数値だけで間に合うようなプログラムはまずありませんので2バイトのレジスタを使って2バイトの演算が出来ることはたいへん助かります。

だいいち、全てのアドレスを表わすには、どう

《第14—A図》Z80の使用可能レジスタ

しても2バイト必要ですから、1バイトのレジスタのみではアドレスを指すこともできません。

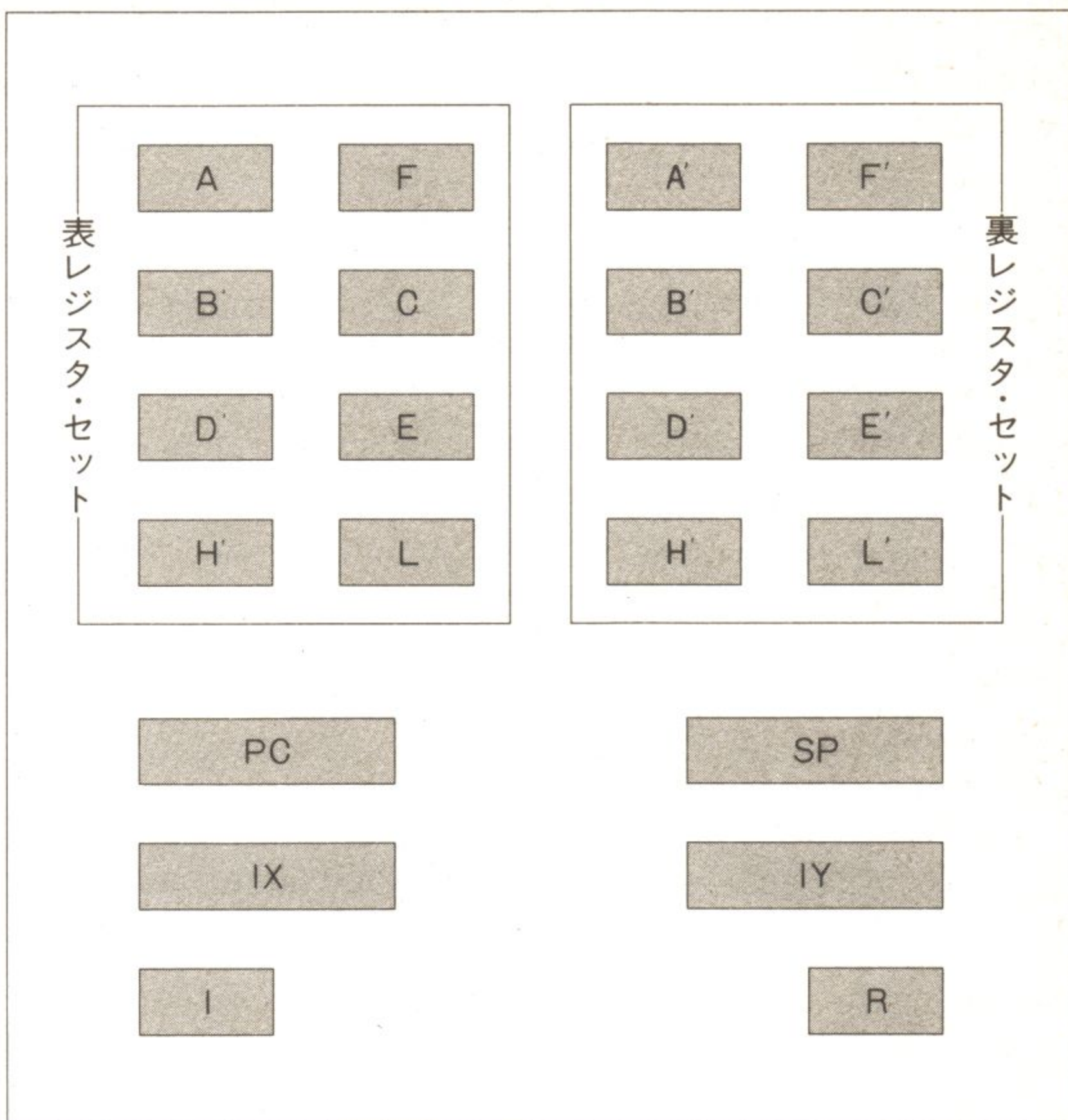
特にHレジスタとLレジスタを対にしたHLレジスタ対は、アドレス修飾のための命令を多く持っているためアドレス修飾の時はHLレジスタ対が最もよく使用されます。又、HLレジスタ対は16ビットの演算の時も中心となります。8ビットの場合アキュムレータがそうであるように、16ビットの演算命令はHLレジスタ対と他のレジスタ対の間で行われる事が多いですし、その場合は結果がHLレジスタ対に入ります。

レジスタ対は二つのレジスタを対にして使うのですから、例えばHLレジスタ対とHレジスタ、Lレジスタを同時に別の用途に使用することはできず、一方の値が変化すれば必然的に他方の値も変化してしまいます。

また、Z80には、A、F、B、C、D、E、H、Lの8個の表レジスタの裏レジスタとして、A'、F'、B'、C'、D'、E'、H'、L'の各レジスタが用意されていますが、表レジスタと裏レジスタを同時に使用することはできません。同時に使用できないものがなぜわざわざ用意してあるのか疑問に思うかもしれませんが、それなりの利用方法があるのです。

例えば全てのレジスタを使用しているメイン・ルーチン中からサブルーチンを呼び出すような場合には、サブルーチンの始めて全てのレジスタの内容をメモリ上に退避しておいて、サブルーチンから抜け出る前に今度はメモリ上に退避しておいた内容をレジスタにもどしておく、などという手法がよく行われますが、この時レジスタの内容をいちいち退避しておくかわりに表レジスタと裏レジスタを交換しておき、メイン・ルーチンにもどる前にもう一度交換しておけばよいのです。つまりサブルーチン中では、裏レジスタだけを使用したことになり、この手法は特に割り込み処理ルーチン内で有効です。

裏レジスタも便利なのですが、残念ながら裏レ



ジスタを十分に活用したプログラムはあまり多くありません。8080CPUの頃には無かったレジスタですし、裏レジスタを多用したプログラムが大へん読みにくくなることも事実ですので始めのうちは8個の表レジスタのみを有効に活用する方法をよく学習し、慣れたところで、サブルーチン中で使い切るような使い方のみに使用することを勧めます。

実際、6800や6502などのCPUに比較すれば8個のレジスタのみでも十分なはずですし、それでも足りなければ、実際のプログラムでは足りない場合がほとんどですが、メモリ上に入れておいて必要な時にだけレジスタに持って来て参照し、またメモリ上に記憶させておくという方法を使えばよいのです。

Z80のプログラム・カウンタ(PC)

プログラム・カウンタ(PC)は、16ビットのレジスタで、常に現在実行しているアドレスの次のアドレスが入っています。

《第14—B図》 Z80の使用可能レジスタ

| レジスタ | 名 称 | ビット バイト | 8080 に有? | 機 能 |
|------|--------------------|------------|-------------|-------------------------|
| PC | プログラム カウンタ | 16 2 | ○ | 現在実行中の次のアドレスを常に指す |
| SP | スタック ポインタ | 16 2 | ○ | スタックの最下位アドレスを指す |
| IX | インデックス レジスタ | 16 2 | × | 8080の時不可能だった相対アドレス指定に使用 |
| IY | インデックス レジスタ | 16 2 | × | |
| I | インタラプト・ベクタ レジスタ | 8 1 | × | モード2の割り込み時に使用 |
| R | メモリ・リフレッシュ レジスタ | 7 | × | ダイナミック・ラムのリフレッシュに使用 |

| 表レジスタ群 | | | | | 裏レジスタ群 | | | | | |
|--------|--------------------|------------|-------------|--------------------------|--------|----------|------------|-------------|--------------------|--|
| レジスタ | 名 称 | ビット バイト | 8080 に有? | 機 能 | レジスタ対 | 名 称 | ビット バイト | 8080 に有? | 機 能 | |
| A | アキュムレータ =Aレジスタ | 8 1 | ○ | Fレジスタ以外は8ビットの汎用レジスタとして使用 | AF | | | | | |
| F | フラグ・レジスタ | 8 1 | ○ | | | | | | | |
| B | B レジスタ | 8 1 | ○ | | BC | BCレジスタ対 | 16 2 | ○ | 16ビットの汎用レジスタ対として使用 | |
| C | C レジスタ | 8 1 | ○ | | | | | | | |
| D | D レジスタ | 8 1 | ○ | | | | | | | |
| E | E レジスタ | 8 1 | ○ | | | | | | | |
| H | H レジスタ | 8 1 | ○ | | HL | HLレジスタ対 | 16 2 | ○ | | |
| L | L レジスタ | 8 1 | ○ | | | | | | | |
| A' | アキュムレータ =A'レジスタ | 8 1 | × | 機能は表レジスタと同じだが、両者は同時に使用不可 | AF' | | | | | |
| F' | フラグ・レジスタ | 8 1 | × | | | | | | | |
| B' | B' レジスタ | 8 1 | × | | BC' | BC'レジスタ対 | 16 2 | × | | |
| C' | C' レジスタ | 8 1 | × | | | | | | | |
| D' | D' レジスタ | 8 1 | × | | | | | | | |
| E' | E' レジスタ | 8 1 | × | | | | | | | |
| H' | H' レジスタ | 8 1 | × | | HL' | HL'レジスタ対 | 16 2 | × | | |
| L' | L' レジスタ | 8 1 | × | | | | | | | |

注) 太線内が特に重要

例えば、現在E 0 0 0 H番地の命令を実行している時点ではプログラム・カウンタには、E 0 0 1が入っていますしE 0 0 1 H番地を実行している時点ではE 0 0 2が入っている、というように命令を実行して行くごとに一つずつ増加していきます。

もし、途中でジャンプ命令などがあればプログラム・カウンタには、飛び先のアドレスが入りますし、リセット・スイッチが押されればプログラム・カウンタに0 0 0 0が入りZ 80は0 0 0 0 H番地から実行を始めます。

現在の所、マイコン誌上などで“PC”と目すればPC-8001のことを思いうかべる方が多いと思いますが、プログラム・カウンタのことも“PC”と書きますから、今後も本文中で“PC”と書いてあったらPC-8001のことではなくプログラム・カウンタのことだと思ってください。

Z 80のスタックポインタ(SP)

スタックポインタも16ビットのレジスタです。スタックポインタの役割りを理解するためにはまず、スタックについて知らなければなりません。詳しくは、レジスタ退避命令(PUSH, POP命令)やサブルーチンコール(CALL命令)の時に登場しますのでここではサブルーチン・コールを例にとってごく簡単に説明します。

サブルーチンと呼び出す時には当然メイン・ルーチンに戻る時のことを考えて戻り先のアドレスを覚えておかなければなりません。これはメモリ上にスタックと呼ばれるエリアをとってメモリの上位から下位へと順々に蓄えられていきます。そしてメイン・ルーチンに戻る必要の出た時に今度は逆にスタックの内容を下位から上位へと参照して戻り番地を取り出すのですが、スタック・ポインタは、このスタックの最下位バイトを常に指しているレジスタなのです。

これだけの説明では、おそらくはつきりしない方が多いと思いますが、レジスタ退避の命令を使うようになればわかってくるはずです。

インデックス・レジスタ (IX・IY)

Z 80になって8080の頃無かった2本のインデックス・レジスタが使えるようになりました。

8080の頃は、多くのデータをメモリ上に置いて扱うような場合には、HLレジスタ対にアドレスを入れておいてアドレス修飾を行っていたわけですが、インデックス・レジスタが加わってより楽になったと言えます。ただしこのレジスタもあれば大へん便利なレジスタですが、もし無かったとしてもプログラムが組めないというような事はありません。

その他のレジスタ

Z 80は他にもインタラプト・ベクタレジスタ(I)と、メモリリフレッシュレジスタ(R)がありますが前者は割り込み機能の拡張にともなって、後者はダイナミック・ラムを直接つなげるために加えられたレジスタですのでこの2本のレジスタを使うためには、ハードを熟知しなければなりません。ここでは説明をはぶかせていただきますが興味がおありの方は参考文献等をさがしてみてください。いずれにしろ初心者が下手に使用できないレジスタです。

レジスタのまとめ

Z 80が持つレジスタ群の数々について長々と説明しましたが全てを一度に理解する必要はありません。

まず始めに覚えておいて欲しいのは、A, B, C, D, E, H, Lの7個の汎用レジスタに各々8ビットの情報(数値)を記憶させて、BASICのいわゆる変数のような使い方ができるということ、BとC, DとE, HとLをそれぞれ対にして3組の16ビット汎用レジスタとして扱うこともできるということぐらいです。

マイコンのための数についての豆知識

マイクロ・コンピュータを勉強して行くと、よくバイトとかビットとかいう言葉が目につきます。特にマシン語を理解するためには、この意味をよく知っておく必要がありますし、この本文中でもかなり多く、登場させてしまいましたので簡単に説明したいと思います。

まず最少の単位がビットで、2進数に直した時の桁数に当たります。コンピュータでは、最終的には1か0かしか無い2進数を扱うのですが人間が読む場合は0と1の列だけでは、わかりにくいので、2進数を4桁ごとに分けるとそれぞれが0～15までの数になります。その0～15までの数に0～Fまでを割りあてて1桁とし、人間にも見易くしたものが現在マシン語などを扱う場合の中心になっている16進数です。

ですから2進数を16進数に変換するには、まず下位から4桁ごとに分けてしまい、それぞれ4桁ずつの束に0～Fまでを割りあててそれを16進数の1桁にすればよいのです。また10進数と16進数を直接変換する事は大へんですから1度2進数を介して変換するようにすればよいでしょう。

また、8ビット＝1バイトとなり、最近のCPUは、Z80も含めて普通は8ビット並列処理となっているため、1バイトごとに、0000H番地からFFFFH番地までのアドレスが割り当てられ

ているのは皆さんも御存知のとおりです。8ビット並列処理と言え、8ビットの数が同時に扱えるということですから2進数では8桁、16進数では2桁、10進数に直せば0～255までの数値が一つのアドレスに割り当てられるということです。

また、Z80には、2バイトの数値を取り扱うための命令も数多く用意されていますが、2バイト(16ビット)のことを1ワードと言うこともあります。

マシン語プログラムやメモリの量などを表わすときなどには、約1024バイトを一つにまとめて1Kバイトなどと言いますから、例えば4.5Kバイトと言え、約4608バイトの事となります。

4608バイトといわれてもどのくらいの量かわかりにくいと思いますが「ダンプ・リストでぎっしり2ページ程度」と言え、見当がつくでしょう。

BASICによる10進↔16進変換

BASICのダイレクト・モードで10進↔16進の変換をすることができます。

例えば10進数の100を16進数に変換するためには、

? HEX\$ (100) CR

で、16進数のA0を10進数に直すには、

? &HA0 CR

を行うだけでよいのです。

《第15図》BEEP, BEEP 1, BEEP 0 の実際のプログラム

| BASIC(標準) | BASIC(変形) | マ シ ン 語 | | |
|---------------------|--|----------------------|----------------------|-----------------------------------|
| | | アドレス | オブジェクト | ニーモニック |
| 10 BEEP 1 20 END | 10 A=&H20 20 OUT&H40, A 30 END | E000 E002 E004 | 3E 20 D3 40 76 | LD A, 20H OUT (40H), A HALT |
| 10 BEEP 0 20 END | 10 A=0 20 OUT&H40, A 30 END | E000 E002 E004 | 3E 00 D3 40 76 | LD A, 00H OUT (40H), A HALT |
| 10 BEEP 20 END | 10 DEFUSR=&HD43 20 I=USR(0) 30 END | E000 E003 | CD 43 0D 76 | CALL 0D43H HALT |

《第16図》音出しのプログラム例

```

100  '*****
110  '*** アメパト サイレン ***
120  '*****
130  CLEAR 300, &HE8FF
140  DEFUSR=&HE900
150  FOR A=&HE900 TO &HE922
160      READ B$
170      POKE A, VAL("&H"+B$)
180  NEXT A
190  C=USR(0)
200  WIDTH,
210  END
220  '*** マシngo データ ***
230  DATA 3E, 00, D3, 51, 1E, 01, 0E, 02
240  DATA 43, 3E, 20, D3, 40, 10, FE, 43
250  DATA 3E, 00, D3, 40, 10, FE, 0D, 20
260  DATA EF, 1C, 20, EA, DB, 09, FE, BF
270  DATA 20, E2, C9

```

Q&Aコーナー

PC-8001は、ハード、ソフト共にすぐれた機能を持っていますが他の同系機種に比較して決定的にももの足りない点は、N-BASICでの音出し機能の不足でしょう。

他の機種を持っている友人達と各機種間の比較論争などを行うと、まっ先にやり玉に上げられるのが、N-BASICに音楽演奏のための命令が無いことではないでしょうか？

もちろんハードを加えればこのような問題はなくなるのですが、ソフトウェアだけでもこの欠点を十分に補うことができるのです。

マシン語による電子オルガンはTK-80などワンボード・マイコンのころからよく行われて来ましたのでここでは、マシン語での内蔵スピーカからの音出し(BEEP1)音消し(BEEP0)と一定時間の音出し(BEEP)のみを紹介します。レジスタの説明からは少しはずれるかもしれませんが、いろいろな応用が可能だと思いますので、学習が終わった時点で、もう一度見直して欲しいと思います。

PCの内蔵スピーカは、出力ポート40Hの第5ビットによってコントロールすることができます。

つまりポート40Hに20Hを出力することによってBEEP1の、00Hを出力することによってBEEP0のそれぞれ代用ができるのです。

出力ポート40Hには、第5ビットのスピーカの他にも幾つかの周辺機器のコントロールに使用されているため、上の方法で勝手に他のビットを変えてしまえば他の動作に支障が起こるはずなのですが、ゲーム等の場合にはほとんど問題ありません。ちなみにN-BASICインタプリタでは、他のビットを変えないように計算しています。

また、一定時間音を出すBEEPのマシン語はN-BASIC内にサブルーチンが用意されていますので、それを利用するのが一番簡単でしょう。

BEEP, BEEP1, BEEP0共に実際のプログラムを第15図に示します。それぞれがBASICの最も標準的な方法、マシン語と同じ方法、マシン語による方法に分けてありますので試してみてください。

サイレン

「ソフトのみで他機種の音楽演奏機能以上のことができる」と言われても「ほんとうかな?」と疑いたくなるのが人間の心理だと思いますので第16図に音出しのプログラム例として“PC-8001の内蔵スピーカーによるアメパト・サイレン”を紹介します。

なお、このプログラムは一応 BASIC のデータになっていて、BASIC から入力、“RUN”を行います。メモリ上に“POKE”された後は立派なマシン語ですから暴走させると取り返しのつかないことになります。短いプログラムですから入力し直しても大したことはないと思いますが、走らせる前に一度テープにセーブしておいた方がよいでしょう。

アメパト・サイレンは、“SPACE”バーを押

すと鳴り止みます。

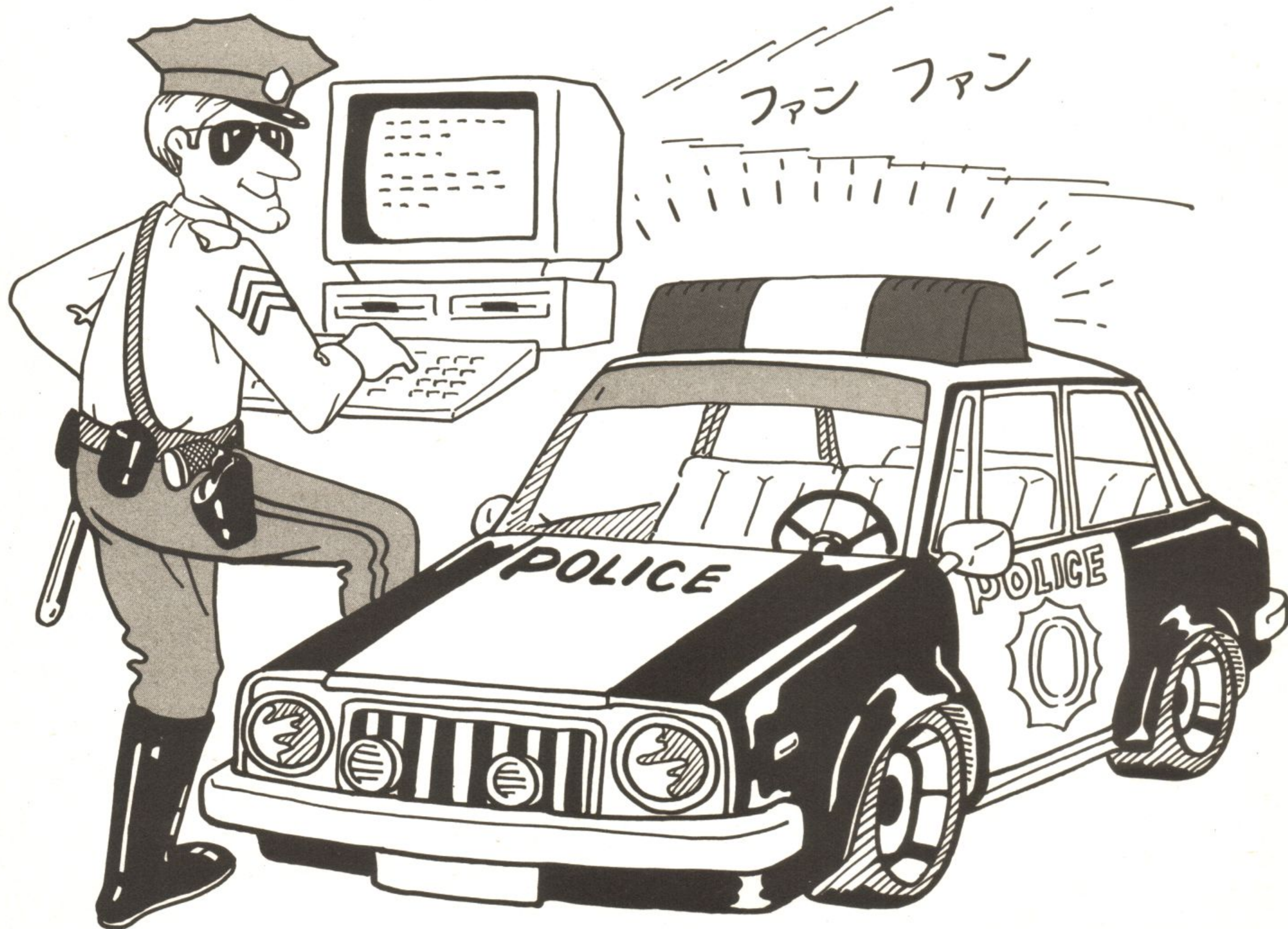
またこのプログラムでは、少しでも音をよくするためには画面を消していますが、230行のマシン語データの始めから4個を00に変更すれば画面は消えなくなりますが、音は雑になります。

マシン語の00はニーモニックでは、“NOP”と書き、“NO OPERATION”の略で、何もしないという事です。

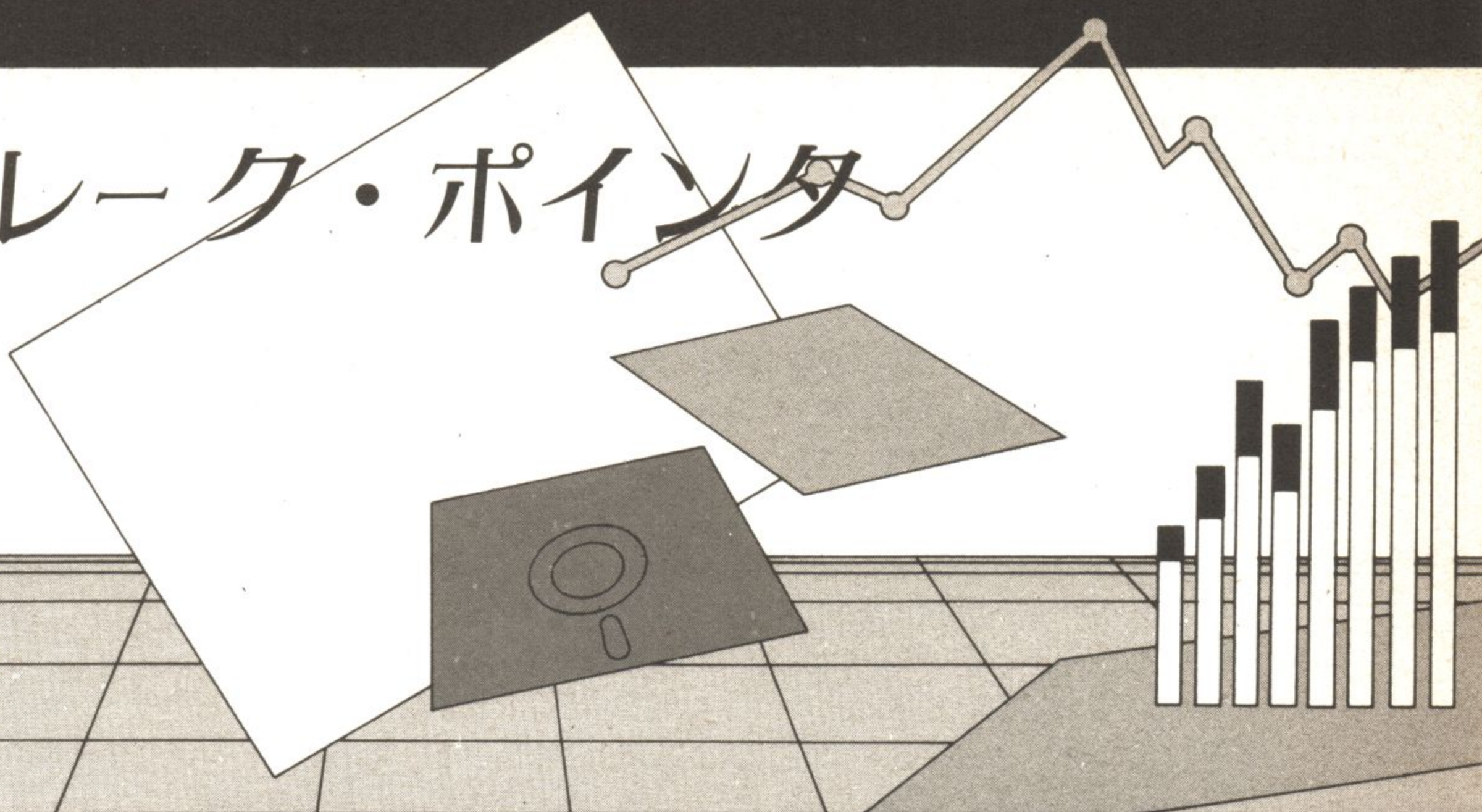
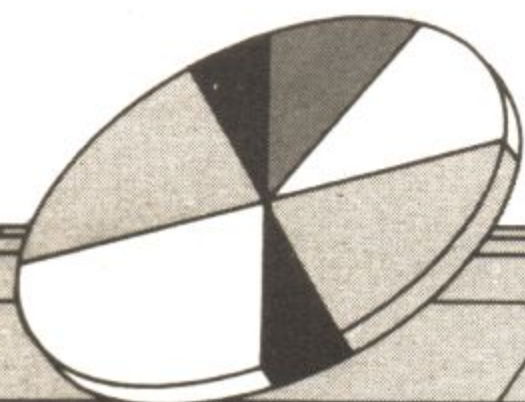
230行最後の02をいろいろな値に変えてみると全体の速度が変化しますので、いろいろな音に変わりますからぜひ試してください。

まとめ

本章ではレジスタ群の簡単な紹介を行いました。次章では第1ブロックの総まとめとして、マシン語学習用のツールを紹介しましょう。



ミニ・ブレイク・ポインタ



前章までの内容で、マシン語とはどんなものか
大方の見当は付けていただけたものと思います。

ここで、本章も含めた第1ブロックすべて、す
なわち第1章から第4章までの簡単な内容を示し
ておきましょう。

第1章 コーディングの実際

PC-8001のV-RAM

ニーモニック (アセンブリ言語)

第2章 アセンブルの実際

PC-8001のメモリ・マップ

N-BASICのCLEARコマンド

アセンブルの実際

マシン語のコーディング書式

BASICとマシン語の組合せ

第3章 Z80のレジスタ・セット

Z80の全レジスタ・セット

マシン語による内蔵スピーカの制御

アメリカン・パトロール・サイレン

第4章 ミニ・ブレイク・ポインタ

ミニ・ブレイク・ポインタ

マシン語によるWIDTHの設定

マシン語によるCONSOLEの設定

も満たない超簡易型モニタで必要最少限の機能し
か持っていないため、どうしてもマシン語を使っ
て行く上では不便なのです。

個々の命令をPC-8001に入力してその結果を
調べてみるためには、最少限レジスタの値ぐらい
は確認できなくてはなりません。

そこで第1ブロックの最終章では、初心者でも
簡単に使用できるレジスタ表示プログラム“MINI
BREAK POINTER”をソース・リスト付きで紹介
し、その使い方を説明します。

入力方法

PC-8001の電源を入れた後“MONCR”でマ
シン語モニタに移し、“S”コマンドを使って第17
図のダンプ・リストを全て入力してください。

チェック・サムなどは用意してありませんが、
短いプログラムですからもう一度見直し、完全に
間違いが無くなった時点で、カセット・テープに
セーブしておいてください。1個所の入力ミスで
も暴走する可能性がありますからよく注意してく
ださい。

さて、第2ブロックからはいよいよZ80に用意さ
れた命令の数々を順を追って解説していくことに
なるわけですが一つ困った事があります。

PC-8001のマシン語モニタが、1Kバイトに

《第17図》 MINI BREAK POINTER ダンプリスト

```
ED00 3A E3 F1 FE C3 CC 6C ED
ED08 3E 4A CD 79 ED E5 CD CA
ED10 5F 3E 50 CD 79 ED 22 94
ED18 ED 7E 32 93 ED 36 FF 3E
ED20 C3 32 E3 F1 21 41 ED 22
ED28 E4 F1 21 22 11 E5 F1 01
ED30 44 33 11 66 55 21 88 77
ED38 DD 21 AA 99 FD 21 CC BB
ED40 C9 FD E5 DD E5 E5 D5 C5
ED48 F5 CD CA 5F 06 06 E1 CD
ED50 C0 5E CD D4 5F 10 F7 E1
ED58 2B CD C0 5E CD D4 5F 21
ED60 00 00 39 CD C0 5E CD 6C
ED68 ED C3 66 5C 3A 93 ED 2A
ED70 94 ED 77 3E C9 32 E3 F1
ED78 C9 CD 57 02 21 00 00 CD
ED80 B9 5F FE 0D C8 CD 39 5E
ED88 38 F5 CD 57 02 CD 4B 5E
ED90 18 ED
```

使用方法

本章の様なプログラムは、普通のゲーム・プログラムなどと違って使い方を熟知しなければ意味がありませんからよく憶えてください。

まず“GED00^CR”を行いプログラムを走らせると“J”というプロンプトが出て入力待ちになります。ここで対象となるプログラムのスタート・アドレスを16進数で入力してやります。16進数関係のキー以外は無視されますが、**STOP** キーではマシン語モニタにもどり **RETURN** キーを押すと、入力が終了となります。

もし間違ったアドレスを入力した場合は、その後続けて新しく入力すれば常に最後からの4桁が有効になります。

スタート・アドレスの入力が終了すると次に“P”のプロンプトが出て再度入力待ちになりますから、ここでは対象となるプログラムのどのアドレスにブレーク・ポインタを設定するかを、やはり16進数で入力します。

ブレーク・ポインタを入力すると、自動的に対象となるプログラムのスタート・アドレス (“J”

の時に入力したアドレス) から実行を始め、ブレーク・ポイントの前で実行を停止してマシン語モニタのコマンド待ち (“*”表示) になりますが、その際に4桁の16進数8個をスペース (空白) をはさんでプリントします。

この8個の16進数が重要なわけで、これがブレーク・ポイントの手前の時点での各レジスタの値なのです。

左から、AF, BC, DE, HL, IX, IY, PC, SPの各レジスタの値が表示されているはずですので、これによって各レジスタに、どんな値が入っているのかがわかるのです。

実際にはAF, BC, DE, HL, I, Rの各レジスタが残っていますが、裏レジスタの値を見たい事はあまり多くありませんし、I, Rの両レジスタも必要ありません。何よりも、横40字モードの時の画面で1行に収めなかったために省略させていただきましたので御了承ください。表示するように改良する事は簡単にできます。

TEST PROGRAM

それでは、第18図のプログラムを例にして実際にブレーク・ポインタを設定してみましょう。

このプログラムは、ブレーク・ポインタのためのテストプログラムで、このプログラムを実行すると各レジスタは次のような値になります。

```
AF = 1 1 2 2 H
BD = 3 3 4 4 H
DE = 5 5 6 6 H
HL = 7 7 8 8 H
IX = 9 9 A A H
IY = B B C C H
SP = F F F F H
```

しかし、PC—8001のモニタからは、このようにレジスタの値が変化したことを確認することができませんので、ブレークポイントを設定して各レジスタの値が本当に上記のようになっているのかを調べてみましょう。

《第18図》 テストプログラム



```

;*****
;  TEST PROGRAM
;*****
;
;          ORG  0C030H
;
C030 212211      LD  HL,1122H
C033 E5          PUSH HL
C034 F1          POP  AF
C035 014433      LD  BC,3344H
C038 116655      LD  DE,5566H
C03B 218877      LD  HL,7788H
C03E DD21AA99    LD  IX,99AAH
C042 FD21CCBB    LD  IY,0BBCCH
C046 31FFFF      LD  SP,0FFFFH
C049 C3665C      JP  5C66H
;
C04C          END

```

まず第18図のテストプログラムのオブジェクト（マシン語）の部分で、C030H～C04BHまでに全て入力してください。

当然先程入力したブレーク・ポインタは入っているものとしてします。

次に“GED00CR”によってブレーク・ポインタを走らせます。

Jの表示には、テスト・プログラムの先頭アドレスである“C030CR”を、Pには、“C049CR”をそれぞれ入力すると、画面に各レジスタの値を表示してマシン語モニタにもどります。

これによって、テスト・プログラムが正常に動作しているのかを確認することができるのです。

この時、PC（プログラム・カウンタ）の値がC049と表示されますがこれによって、前章で説明した、プログラム・カウンタは常に現在実行している次のアドレスを指しているということを、実際に理解していただけたのではないかと思います。C049H番地にブレーク・ポイントを設定したという事は、C048H番地までを実行することですからその時点でプログラム・カウンタは、C049H番地を指しているわけなのです。

BREAK POINTERの注意

ブレーク・ポインタは、ED00H番地からの約150バイトに位置しています。皆さんもお気づきだと思いますが、ここは第2章で説明したフリー・エリアではないのです。そこで利点と欠点が生まれるわけですが、まず利点としては、フリー・エリアのどの部分に位置しているマシン語のプログラムに対してもブレーク・ポイントを設定することができる点です。

欠点として注意して欲しいのは、EC96H～ED95H番地はN—BASICの入力バッファなので約100文字以上からなる行をBASICから入力、修正すると、ブレーク・ポインタが壊されてしまう点で、そのような時は再度ロードしなければなりません。もちろんマシン語のみのプログラムを使っていれば問題ありません。

またこのプログラムでは、ROM上のプログラムにブレーク・ポイントを設定することはできませんが、これは、ブレーク・ポイントのアドレスに、FFHを書き込むことによって、レジスタの表示ルーチンへジャンプさせているからです。

各自による変更方法

このプログラムでは、入力時のプロンプトとしてJumpの“J”とbreak Pointの“P”を使用しましたが、それぞれのキャラクタコードがED09H番地とED12H番地に入っていますので、ここに別のキャラクタコードを入れることによって簡単に変更することができます。

この他にもいろいろな改良点、改造点があると思いますので、ためしてみてください。

Q&Aコーナー

このコーナーでは、マシン語による画面の初期設定の方法、つまりWIDTHやCONSOLEを切り換える方法を説明します。

何かむずかしそうに思えるかもしれませんがN—BASIC ROM内のサブルーチンを多用してできる限り簡単にまとめましたのでぜひ試してください。

◎WIDTHの切り換え方

まず、WIDTH⑦、①に切り換える場合を例にとって以下に手順を示します。

- ①、Bレジスタに⑦を入れる。
- ②、Cレジスタに①を入れる。
- ③、093AHからのサブルーチンを呼ぶ。

これだけでよいのです。①、②ではLD命令を使って代入すればよく、③では、CALL命令によってサブルーチンを呼べばよいのです。

具体的なプログラム例として、WIDTH 40, 25に切り換えるプログラムを第19図に上げておきます。

◎CONSOLEの切り換え方

CONSOLE ⑦、①、⑤、⑥を切り換える場合の手順は、

- ①、EA5EH番地に、⑦+1を入れる
- ②、EA5DHに、①-1を入れる
- ③、Bレジスタに、⑤を入れる(0又はFFH)
- ④、Cレジスタに、⑥を入れる(0又はFFH)
- ⑤、08F7 Hからのサブルーチンを呼ぶ

となります。この場合③、④で言う⑤、⑥はBASICの時の0, 1とは違っていますので注意が必要です。BASICで0の時は、0でよいのですが

《第19図》WIDTH40, 25を切り換える一例

| アドレス | オブジェクト | ニ ー モ ニ ッ ク | コ メ ン ト |
|------|----------|-------------|-------------------|
| C100 | 06 28 | LD B, 28H | Bレジスタに40を入れる |
| C102 | 0E 19 | LD C, 19H | Cレジスタに25を入れる |
| C104 | CD 3A 09 | CALL 093AH | 093AHからのサブルーチンを呼ぶ |
| C107 | C3 66 5C | JP 5C66H | マシン語モニタにジャンプする |

《第20図》CONSOLE 0, 25, 0, 1を切り換える一例

| アドレス | オブジェクト | ニ ー モ ニ ッ ク | コ メ ン ト |
|------|----------|---------------|-------------------|
| C100 | 3E 01 | LD A, 01H | } EA5EH番地に1を入れる |
| C102 | 32 5E EA | LD (EA5EH), A | |
| C105 | 3E 18 | LD A, 18H | } EA5DH番地に24を入れる |
| C107 | 32 5D EA | LD (EA5DH), A | |
| C10A | 06 00 | LD B, 00H | Bレジスタに00Hを入れる |
| C10C | 0E FF | LD C, FFH | CレジスタにFFHを入れる |
| C10E | CD F7 08 | CALL 08F7 | 08F7Hからのサブルーチンを呼ぶ |
| C111 | C3 66 5C | JP 5C66H | マシン語モニタにジャンプする |

BASICで1の時はFFHとなります。

第20図に、CONSOLE 0, 25, 0, 1に切り換える場合のプログラムをあげておきます。

以上、N-BASIC ROM内のサブルーチンを使えば比較的簡単に画面設定ができるのですが、幾つか注意しなければならない点があります。

間違ったパラメータを渡した場合、例えば、WIDTH 90, 130やCONSOLE 90, 30, 5, 7等を実行してしまった時にどうなるかは保障されません。暴走の危険すらありますから、このようなことが無いように十分なチェックを行ってください。パラメータを省略した時などもよく上記のような状態になることがありますので省略することできません。

◎マシン語モニタへのジャンプ方式

第18図、第19図、第20図のプログラムの最後に“JP 5 C66H”で5 C66H番地にジャンプしていますが、5 C66H番地はマシン語モニタのスタート・アドレスですので、プログラムの最後にこの

命令を書いておけばマシン語モニタのコマンド待ちになって便利です。

前章までは“HALT”命令を使っていましたが、リセット・スイッチを押すのは精神衛生上もよくないので今後、プログラムの最後は、モニタにもどすようにしました。

ちなみにN-BASICからUSR関数によって呼び出す場合は前章の様にRET命令(C9)を使ってください。

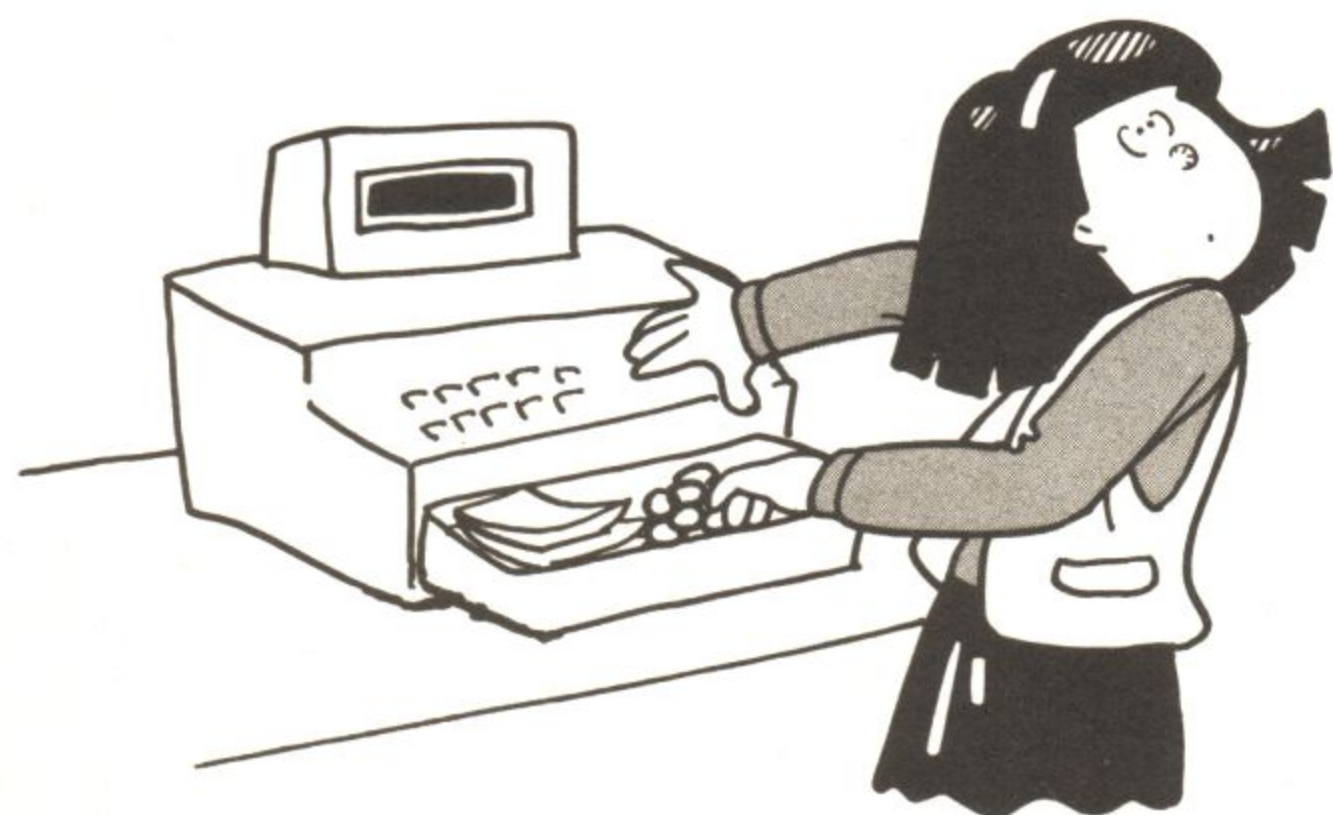
まとめ

マシン語に限らずプログラムを作れるようになるためには自分で個々の命令を試してみることだと思います。どんなプログラムでも結構ですからぜひ自分のPC-8001に入力して走らせ、ブレーク・ポイントを設定してみてください。絶対マシン語に対する興味が倍增することを確認しています。



レジスタの表示には
MINI BREAK POINTER
が便利です

| AF | BD | DE | HL | IX | IY | SP |
|------|------|------|------|------|------|------|
| 1122 | 3344 | 5566 | 7788 | 99AA | BBCC | FFFF |



《第21図》 MINI BREAK POINTER ソースリスト

```

;*****
;  MINI BREAK POINTER Ver 7.0
;*****
;
;      ORG  0ED00H
;
0257      DSPCHR:EQU  0257H      ; display a character
5FB9      INCHR: EQU  5FB9H      ; wait and input a character
5C66      MONJMP:EQU  5C66H      ; entry address of monitor
5E39      HEXCHK:EQU  5E39H      ; check hexa code or not
5E4B      BINCV4:EQU  5E4BH      ; convert and shift binary code
5EC0      HEXOUT:EQU  5EC0H      ; display address
5FCA      CRLF:  EQU  5FCAH      ; display cr code and lf code
5FD4      SPCOUT:EQU  5FD4H      ; display a space
;
ED93      BKWAIT:EQU  0ED93H     ; content of break point
ED94      BPWAIT:EQU  0ED94H     ; address of break point
;
F1E3      RSTART:EQU  0F1E3H     ; fook address restart
;
;-----
;  input address section
;-----
;
ED00 3AE3F1      LD  A,(RSTART)
ED03 FEC3        CP  0C3H
ED05 CC6CED      CALL Z,RESET
;
ED08 3E4A        LD  A,'J'
ED0A CD79ED      CALL HEX4IN
ED0D E5          PUSH HL
ED0E CDCA5F      CALL CRLF
ED11 3E50        LD  A,'P'
ED13 CD79ED      CALL HEX4IN
;
ED16 2294ED      LD  (BPWAIT),HL
ED19 7E          LD  A,(HL)
ED1A 3293ED      LD  (BKWAIT),A
ED1D 36FF        LD  (HL),0FFH
;
ED1F 3EC3        LD  A,0C3H
ED21 32E3F1      LD  (RSTART),A
ED24 2141ED      LD  HL,MAIN
ED27 22E4F1      LD  (RSTART+1),HL
;
ED2A 212211      LD  HL,1122H
ED2D E5          PUSH HL
ED2E F1          POP  AF
ED2F 014433      LD  BC,3344H
ED32 116655      LD  DE,5566H
ED35 218877      LD  HL,7788H
ED38 DD21AA99    LD  IX,99AAH
ED3C FD21CCBB    LD  IY,0BBCCH
;
ED40 C9          RET
;
;-----
;  display registers section
;-----
;
ED41 FDE5      MAIN:  PUSH IY
ED43 DDE5      PUSH IX
ED45 E5        PUSH HL

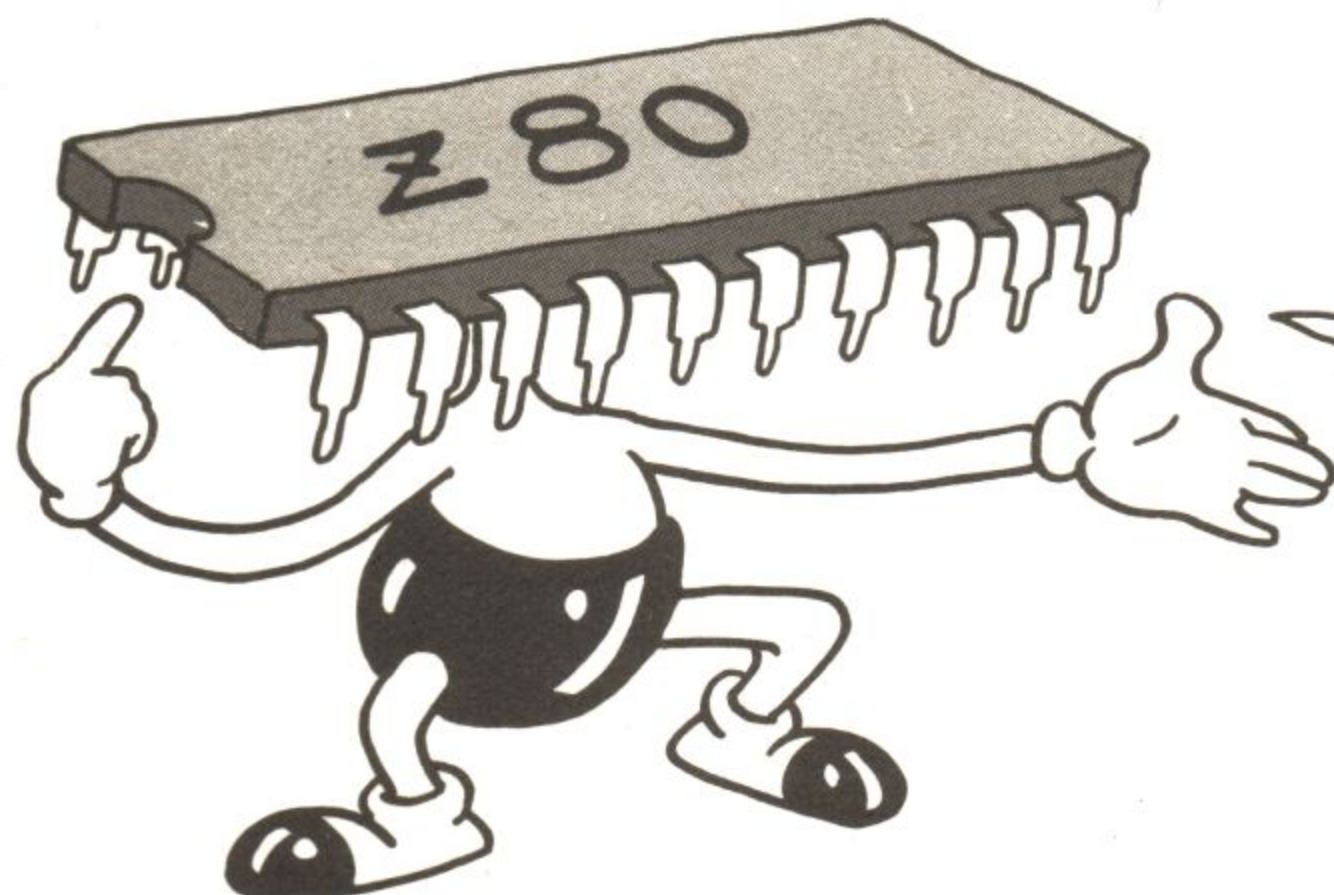
```



```

ED46 D5          PUSH DE
ED47 C5          PUSH BC
ED48 F5          PUSH AF
;
ED49 CDCA5F      CALL CRLF
ED4C 0606        LD B,6
ED4E E1          LOOP: POP HL
ED4F CDC05E      CALL HEXOUT
ED52 CDD45F      CALL SPCOUT
ED55 10F7        DJNZ LOOP
ED57 E1          POP HL
ED58 2B          DEC HL
ED59 CDC05E      CALL HEXOUT
ED5C CDD45F      CALL SPCOUT
ED5F 210000      LD HL,0
ED62 39          ADD HL,SP
ED63 CDC05E      CALL HEXOUT
;
ED66 CD6CED      CALL RESET
ED69 C3665C      JP MONJMP
;
;-----
; subroutines
;-----
;
; clear the break point subroutine
;
ED6C 3A93ED      RESET: LD A,(BKWAIT)
ED6F 2A94ED      LD HL,(BPWAIT)
ED72 77          LD (HL),A
ED73 3EC9        LD A,0C9H
ED75 32E3F1      LD (RSTART),A
ED78 C9          RET
;
; input address subroutine
;
ED79 CD5702      HEX4IN:CALL DSPCHR
ED7C 210000      LD HL,0
ED7F CDB95F      NEXTIN:CALL INCHR
ED82 FE0D        CP 0DH
ED84 C8          RET Z
ED85 CD395E      CALL HEXCHK
ED88 38F5        JR C,NEXTIN
ED8A CD5702      CALL DSPCHR
ED8D CD4B5E      CALL BINCV4
ED90 18ED        JR NEXTIN
;
ED92            END

```



第2ブロックからは
 私とお話するための
 “ことば”を
 覚えて下さい！

CPU マイクロプロセッサの流れ

専用機の時代

マイクロプロセッサの誕生

最も古いマイクロプロセッサ（CPU）は、4ビットの4004（i4004）だといわれています。4004は、インテル社が電卓用に設計して、1971年12月に発表されましたが、このCPUが我国の中小企業であったビジコン社によって発案された事実も有名です。

しかし、この4004も突如として世にでたわけではありません。アメリカの半導体メーカーでは、早くから論理回路中にプログラムを置いて、いわゆるコンピュータのような使い方をしようとする動きが進行していました。実際に、このような動きからは、フェアチャイルド社のPPS-25という電卓向けのLSIセットが生まれています。

このPPS-25は、25ビットのレジスタを4本並列に処理することができた上、アセンブラやシミュレータ等の開発システムも十分に完備しており、現在のマイクロコンピュータのはしりともいえる性格を有していました。それにもかかわらず途中でその座をインテル社の製品群にゆずってしまったのは、各レジスタがBCDコード表現になっているなどの理由によって電卓用以外の使用目的を全く望めなかったことが原因なのでしょう。簡単にいえば、発展性がなかったということです。

事実PPS-25は、我国を中心として数社の電卓に用いられた後、間もなく市場から姿を消しました。しかし、PPS-25のアーキテクチャは、その後しばらくの間はヒューレットパッカード社の関数電卓用LSIセットに受け継がれていました。

これに対して、先の4004は、BCDコードの1桁を処理するのが精一杯の4ビット並列処理です。ハードウェア面

から見たアーキテクチャも欠点だらけで、PPS-25と比べたら、まるで玩具のような代物といってもよいでしょう。

したがって、4004では4ビットを越える演算処理に関しては、すべてソフトウェアによって面倒を見なければなりません。これは、Z80で16ビットを越える加算を行う場合に、キャリー・フラグの内容に注意しながら少しずつ加算を行っていくようなものです。

しかし、このことが、かえって4004に融通性を持たせました。

当初インテル社は、ビジコン社との契約があったため、4004を含むLSIセット（MCS-4）を電卓以外の用途にのみ使用するという条件で市販しました。

4004は従来のLSIでは考えられなかった融通性を認められ、ソフトウェア次第では何でもできるというすばらしい評価を受けたのです。

8008

次にインテル社は、コンピュータの端末用として8ビットの8008（i8008）を開発しました。

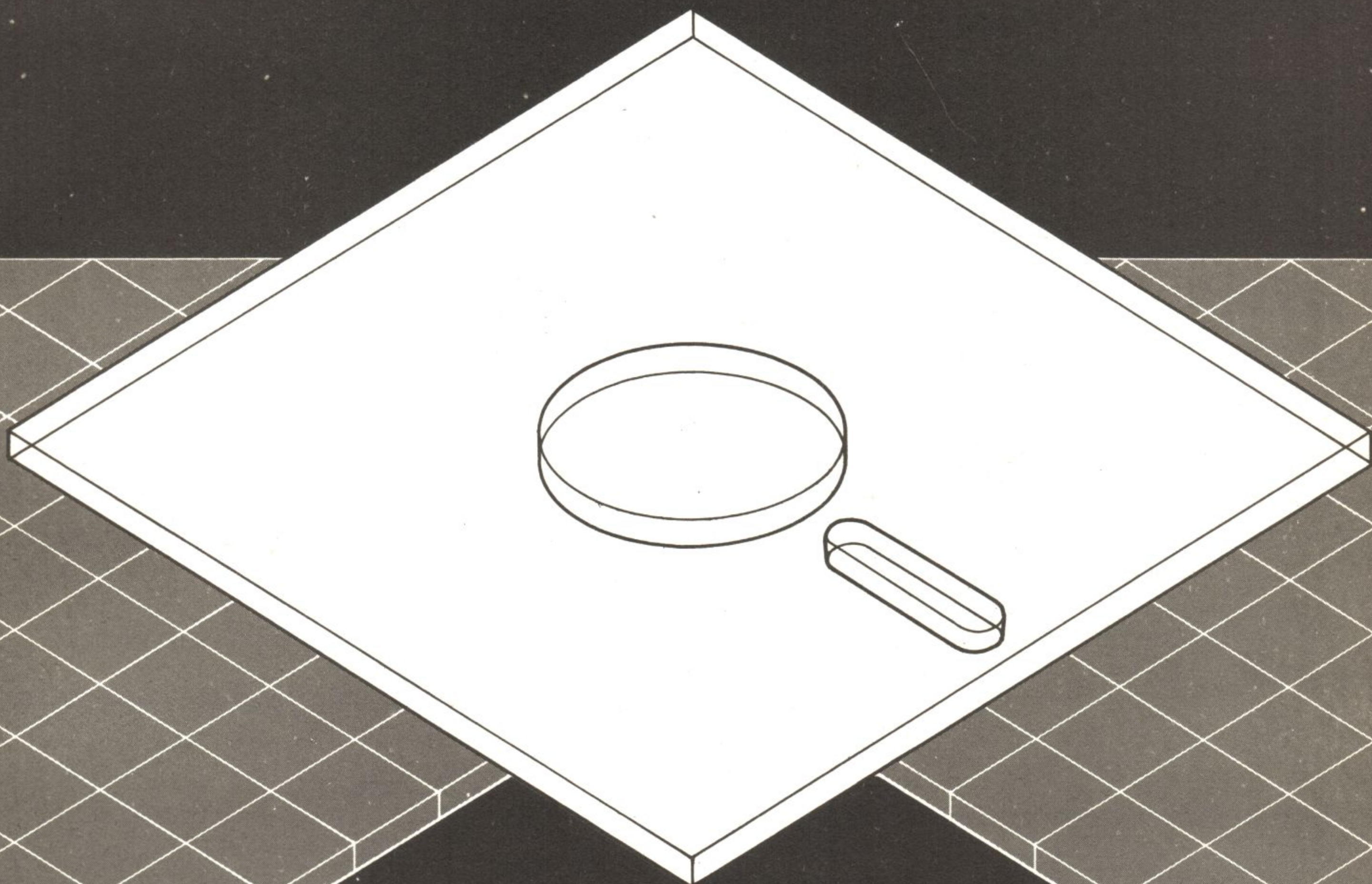
8008は、米データポイント社というコンピュータの端末機を製造している会社の依頼でインテル社が開発を進めていたのですが、データポイント社が要求していた実行速度に比較して比常に低速であったため同社は採用をあきらめました。

そこで、途中から我国の精工舎が自社で製造しているパーソナル・コンピュータに、8008を採用したのです。

こうして、8008は最初の8ビットCPUとして世に出されたのですが、主に実行速度と汎用性の問題から十分な満足が得られるまでにはいたりませんでした。

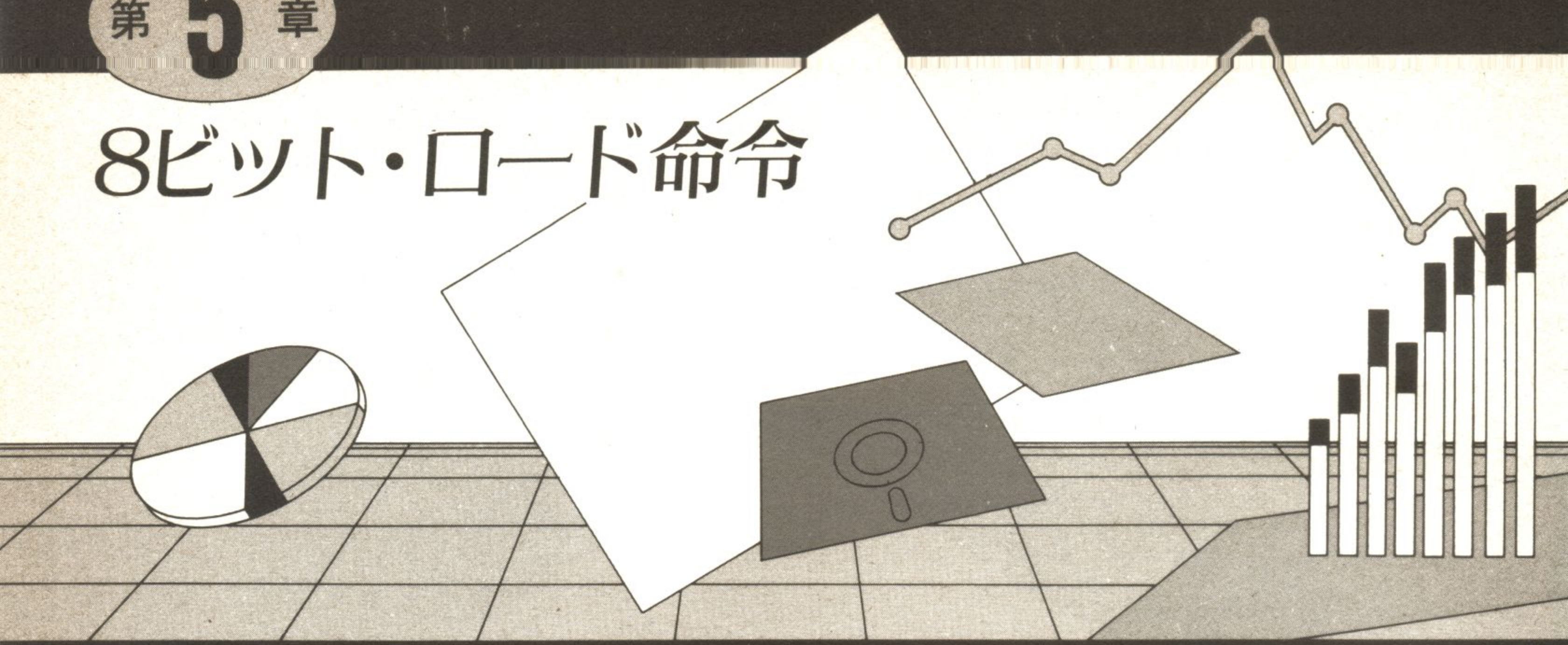
| インテル社 | ザイログ社 | DEC社 | モトローラ社 | MOS テクノロジー社 | |
|-------|-------|------|--------|----------------|--------------------|
| 4004 | | | | | 4ビットCPU 8ビットCPU |
| 8008 | | | | | |

第 2 ブロック



Z80の
インストラクション・セットI

8ビット・ロード命令



第2ブロックからはZ80に用意されたすべての命令を解説しますが、本章ではBASICのLET命令に当たるロード命令，その中でも8ビット（1バイト）の数値のみを扱うための8ビット・ロード命令を紹介します。

ロード命令は基本となる命令ですので簡単な命令ですが，よく覚えておいて欲しいと思います。又，ロード命令さえわかれば，あとの命令はそのバリエーションであると言っても過言ではないと思います。

8ビットロード命令

◎1バイトのレジスタ間で

8ビットの情報を送る命令

BASICでは，

LET C=D (LETは普通省略)

が，この命令に当たり，単にレジスタ間で数値を送るだけのロード命令で，この種の命令の内でも最も単純な部類に入る命令です。ニーモニックは，

LD 第1オペランド，第2オペランド

の書式で書き，第2オペランドの内容がそのまま第1オペランドに代入され，各オペランドには，A，B，C，D，E，H，Lの各レジスタを入れる事ができます。例えば，

LD B，C

とすればCレジスタに入っていた数値がBレジスタに入ります。もちろんCレジスタの内容は変化しませんし，他のレジスタにも全く影響を与えません。

LD A，A や LD C，C

のような意味のない命令も実行できますがこれは，CPUを作る過程で発生してしまったもので時間かせぎの時ぐらいにしか使用されません。

◎レジスタに8ビットの数値を直接代入する命令

この命令も前の命令と同じ様に，

LD 第1オペランド，第2オペランド

という書式で書きます。もともとロード命令には，情報を送る側と受け取る側が必要ですから，どのロード命令にもオペランドは，二つ存在します。

この命令の場合の第1オペランドには，A，B，C，D，E，H，Lのどれかのレジスタが入り，第2オペランドには，第1オペランドに書いたレジスタに代入したい8ビットの数値を直接書きます。例えばAレジスタに16進数の55Hを代入したければ，

LD A，55H

となります。アセンブルするときには，必ず2バイトの命令になり，2バイト目には，第2オペランドをそのまま割り当てます。上の命令をE000H番地からにアセンブルすると，

E000 3E 55 LD A，55H
(アドレス オブジェクト ニーモニック)

となります。

◎HLレジスタ対によるアドレス修飾

N-BASICなどを使ってプログラムを作っている場合には、まず変数が足りなくなることは有りません。しかし、TINY BASICが登場した時には、A～Zまでの26個しか変数が使えず配列が取れないものもあり、簡単なプログラムなどを組む時にも変数が足りなくならないように気を使わなければならない、変数表などを作ってからプログラムを組み始めたものです。

まして、マシン語のレジスタを変数の変わりに使おうと思えば、表レジスタを考えると、A, B, C, D, E, H, Lの7個、2バイトのレジスタ対として考えれば、BC, DE, HLの3組しか使用できないことになります。

このように考えると、変数＝レジスタ(対)と考えて来たのは間違いであった事に気付くはずですが、確かに、レジスタは、変数と同じ働きをしますが、数が全然足りないのです。

短いテスト・プログラムなどを組む時には、レジスタのみで足りる事もありますが、通常のプログラムの場合は、変数(単純変数・配列変数)は、全てメモリ上に置いておくのです。もし、その数値が必要になった場合には、メモリ上からレジスタへ持って来て、参照するなり計算を行うなりした後、またメモリ上へもどしておきます。このように変数を置いてある領域を通常ワーク・エリアなどと言って、プログラム・エリアと区別します。

話は変わりますが、マシン語を使ってテレビ画面上にキャラクタを表示したり、グラフィックで何かのパターンなどを描く場合には、ビデオ・ラム上にそれなりのキャラクタ・コードなり、グラフィック・パターンなりを転送しなければなりません。

以上の点のみを考えても、レジスタ間で数値を扱うだけでは問題があり、どうしてもメモリとレジスタの間で数値の転送を行う命令が不可欠な事がわかっていただけたと思います。しかもその転送はかなり能率的に行えなければなりませんので前回までのように、

```
LD (F300H), A
```

という命令では、そのたびごとにアドレスを指定しなければなりませんから不便です。

そこで考えられるのが、2バイトのレジスタ対を使ってメモリを指定する方法です。どのレジスタ対をアドレス指定専用を使うかですが、最も2バイトの数値に関しては機能の豊富なHLレジスタ対が、8080CPUの頃からよく使われます。

上記のF300H番地にAレジスタの内容を送るための命令を、HLレジスタ対をアドレス指定に使用してプログラムにすると、

```
LD HL, F300H
```

```
LD (HL), A
```

の、2ステップの命令になります。先程の1ステップの命令に対してわざわざ2ステップにして複雑にしたように思えるかもしれませんが、後者の方が全然応用範囲が広がった事に注目してください。

例えば、F300H番地からの10バイトにAレジスタの内容を送りたい時などは、前の方法だと同じような命令を10個続けて使わなければなりません。後者の方法ならBASICのFOR～NEXTループのようなものを作って、HLレジスタ対の値を一つずつ増加させながら次々に代入して行く事も可能です。

以上の様にHLレジスタ対を使ってアドレスを指定し、そのメモリと各レジスタの間で1バイトの数値を転送する事ができるのですが、以下の書式でニーモニックを書きます。

```
LD 第1オペランド, (HL)
```

```
LD (HL), 第2オペランド
```

上の場合は、HLレジスタ対で指定するメモリから第1オペランドに書いたレジスタへの転送、下の場合はその逆で第2オペランドに書いたレジスタからメモリへの転送です。(HL)以外のオペランドには、A, B, C, D, E, H, Lの各レジスタが使えますが、他に16進の数値を直接メモリ上へ送る事ができます。

二～三例を上げますと、

```
LD C, (HL).....①
```

```
LD (HL), E.....②
```

```
LD (HL), 55H.....③
```

などがあります。

①の場合には、HLレジスタ対が指すメモリの

内容がCレジスタに送られますが、ⓐは、その逆にEレジスタの値がHLレジスタ対が指しているメモリに送られます。ⓑが、メモリ上に直接数値を送る例で、この場合は16進数の55Hが送られ、この場合だけはアセンブルする際に2バイトの命令とし、2バイト目に第2オペランドの数値（この例では55H）を割り当てます。

HLレジスタ対をアドレス指定に使う時注意する事は、HLレジスタ対には事前にアドレスを入れておかなければならない点です。

LD HL, アドレス

を使えばよいのですが、これを忘れると関係ないアドレスとの間で数値を転送してしまう事になります。

◎HLレジスタ対以外によるアドレス修飾

Z80では、HL以外のレジスタ対、例えばBCレジスタ対やDEレジスタ対を使ってアドレスを指定することができますが、その場合はAレジスタとの間でしかロード命令が実行できません。

```
LD (DE), A
```

```
LD A, (BC)
```

などだけが使用できます。

又、8080時代には無かったIX、IYのインデックスレジスタをHLレジスタ対のように使って、より便利にメモリ上の数値群を扱うこともできますが、アセンブルをする時点で多少テクニックが必要なため、本章では省略させていただきます。興味のある方やアセンブラを使用している方は一度調べてみるとよいでしょう。

◎他のロード命令

他にもAレジスタとI、Rの各レジスタの間のロード命令もありますが、この命令を使うことはほとんど無いと思いますので、この命令の存在ぐらいを知っておけばよいと思います。

8ビットのロード命令のみを使ったプログラム例

それでは今説明した8ビットのロード命令を使って幾つかのプログラムを組んでみることにしま

しょう。

ロード命令のみではループ処理ができませんので、大した事はできませんが二つ程例をあげておきます。

前章で紹介した“MINI BREAK POINTER”などを使えば、各レジスタ内の値を直接調べることが可能なのですが、まだ入力していない方もいるかもしれませんので、結果は出来る限りメモリ上か画面上（これもV-RAM上ですが）に出して確認できるようにしました。

「マイコンシ」表示プログラム

このプログラムは、画面左上から、“マイコンシ”と表示するためのプログラムです。ただし80字×25行モードにしていけない場合は、一文字おきに、“マコシ”と表示され、“イ”と“ン”が表示されませんので80字モードとし、COLORも確認できる値に設定して画面をクリアしておいてから実行してください。

このプログラム全体では、F300H～F304H番地の5バイトに、マシのキャラクタ・コードをロードすることによって、表示を実現しています。何度もくり返し説明しましたがPC-8001のF300H番地はV-RAMの開始アドレスになっていて、そこから書き込まれたキャラクタ・コードに対応したキャラクタがテレビ画面上に表示されます。

それでは、第22図を行を追ってみて行きましょう。まず、

```
LD A, CFH
```

によってAレジスタに16進数のCFHを代入していますが、このCFHは“マ”のキャラクタ・コードです。キャラクタ・コードを調べるには、N-BASICリファレンス・マニュアルや第3図を参照してください。アセンブラなどを使ってアセンブルする場合には、わざわざキャラクタ・コードを調べなくても、

```
LD A, "マ"
```

```
LD A, 'マ'
```

などと入力すれば自動的に16進数のキャラクタ・コードに変換されるものもあります。

《第22図》「マイコンシ」表示

| アドレス | オブジェクト | ニーモニック | コメント |
|--------------|----------------|----------------------------|---------------|
| D000 | 3E CF | LD A, CFH | A←'マ' |
| D002 | 06 B2 | LD B, B2H | B←'イ' |
| D004 | 0E BA | LD C, BAH | C←'コ' |
| D006 | 16 DD | LD D, DDH | D←'ン' |
| D008 | 1E BC | LD E, BCH | E←'シ' |
| D00A | 32 00 F3 | LD (F300H), A | マ |
| D00D D010 | 21 01 F3 70 | LD HL, F301H LD (HL), B | イ |
| D011 D014 | 21 02 F3 71 | LD HL, F302H LD (HL), C | コ |
| D015 D018 | 21 03 F3 72 | LD HL, F303H LD (HL), D | ン |
| D019 D01C | 21 04 F3 73 | LD HL, F304H LD (HL), E | シ |
| D01D | C3 66 5C | JP 5C66H | モニタへ ジャンプ! |

— キャラクタ→キャラクタ・コード対応表 —

| | | | |
|-------|-------|-------|-------|
| マ→CFH | イ→B2H | コ→BAH | ン→DDH |
| シ→BCH | テ→C3H | ハ→CAH | |
| 。→DFH | フ→CCH | ヤ→ACH | |

話が少しそれましたが、今Aレジスタに“マ”のキャラクタ・コードであるCFHを入れたように、B、C、D、Eの各レジスタにそれぞれ“イ”、“コ”、“ン”、“シ”のキャラクタ・コードである、B2H、BAH、DDH、BCHを代入します。

LD (F300H), A

によって、F300H番地にAレジスタの内容を移していますが、これでV-RAMの先頭番地にCFHが入り、テレビ画面の左上に“マ”というキャラクタ（文字）が表示されたことになります。

その後からは、HLレジスタ対をアドレス指定に使用しています。HLレジスタ対にF300Hを入れた後、今度はHLレジスタ対が指すアドレス(=F301H番地)にBレジスタの内容を移していますが、これでF301H番地に“イ”のキャラクタ・コードであるB2Hが入りテレビ画面には、“マ”に続いて“イ”の文字が表示された事になります。

あとは、“イ”を表示したのと全く同じ方法で次々に残りの“コンシ”をV-RAM上に送り込んでやります。

ジャンプ JP 5C66H

で、マシン語モニタのコマンド待ちにもどっています。

かなり長々と説明しましたが、実際はほんの一瞬の間です。もっともBASICで、“マイコンシ”とプリントさせても「あっ！」という間にプリントしますからマシン語で表示させれば少なくともBASICよりは速いはずですね。

「デンパシンブンシャ」 表示プログラム

第22図では、HLレジスタ対によるアドレス指定や他のレジスタの機能を確認するために、Z80の全表レジスタ・セットをフルに使用しましたが、今度は同じような内容のプログラムをAレジスタだけを使って組んでみました。

第23図を実行させるとテレビ画面上に、

テ “ンパ” シンフ” ンシャ

と表示しますが、これもV-RAM上にキャラクタ・コードを送っているだけです。

このプログラムでは、Aレジスタのみしか使いませんので、キャラクタ・コードをAレジスタに入れたらV-RAM上の適当な位置に転送するという事を文字数回くり返しています。おそらく第22図よりわかり易いプログラムになっているのではないのでしょうか？

一応参考のために同じプログラムを40字モード用に変更したものを第24図としてあげておきます。F300H→F302H→F304HとV-RAMを2番地おきに使っているのがよくわかると思います。

まとめ

以上8ビットのロードを紹介しましたがいかがだったでしょうか。

また、ここで紹介したプログラム例ですが、この他にもいろいろな応用が可能だと思います。ロード命令の復習も兼ねて自分の名前や住所、好きな絵などを画面に表示させてはいかがでしょうか

《第23図》

「デンパシンブンシャ」表示

| アドレス | オブジェクト | ニーモニック | コメント |
|--------------|-------------------|----------------------------|---------------|
| D000 D002 | 3E C3 32 00 F3 | LD A, C3H LD (F300H), A | テ |
| D005 D007 | 3E DE 32 01 F3 | LD A, DEH LD (F301H), A | " |
| D00A D00C | 3E DD 32 02 F3 | LD A, DDH LD (F302H), A | ン |
| D00F D011 | 3E CA 32 03 F3 | LD A, CAH LD (F303H), A | ハ |
| D014 D016 | 3E DF 32 04 F3 | LD A, DFH LD (F304H), A | 。 |
| D019 D01B | 3E BC 32 05 F3 | LD A, BCH LD (F305H), A | シ |
| D01E D020 | 3E DD 32 06 F3 | LD A, DDH LD (F306H), A | ン |
| D023 D025 | 3E CC 32 07 F3 | LD A, CCH LD (F307H), A | フ |
| D028 D02A | 3E DE 32 08 F3 | LD A, DEH LD (F308H), A | ゝ |
| D02D D02F | 3E DD 32 09 F3 | LD A, DDH LD (F309H), A | ン |
| D032 D034 | 3E BC 32 0A F3 | LD A, BCH LD (F30AH), A | シ |
| D037 D039 | 3E AC 32 0B F3 | LD A, ACH LD (F30BH), A | ヤ |
| D03C | C3 66 5C | LD 5C66H | モニタへ ジャンプ! |

《第24図》 40文字モード用

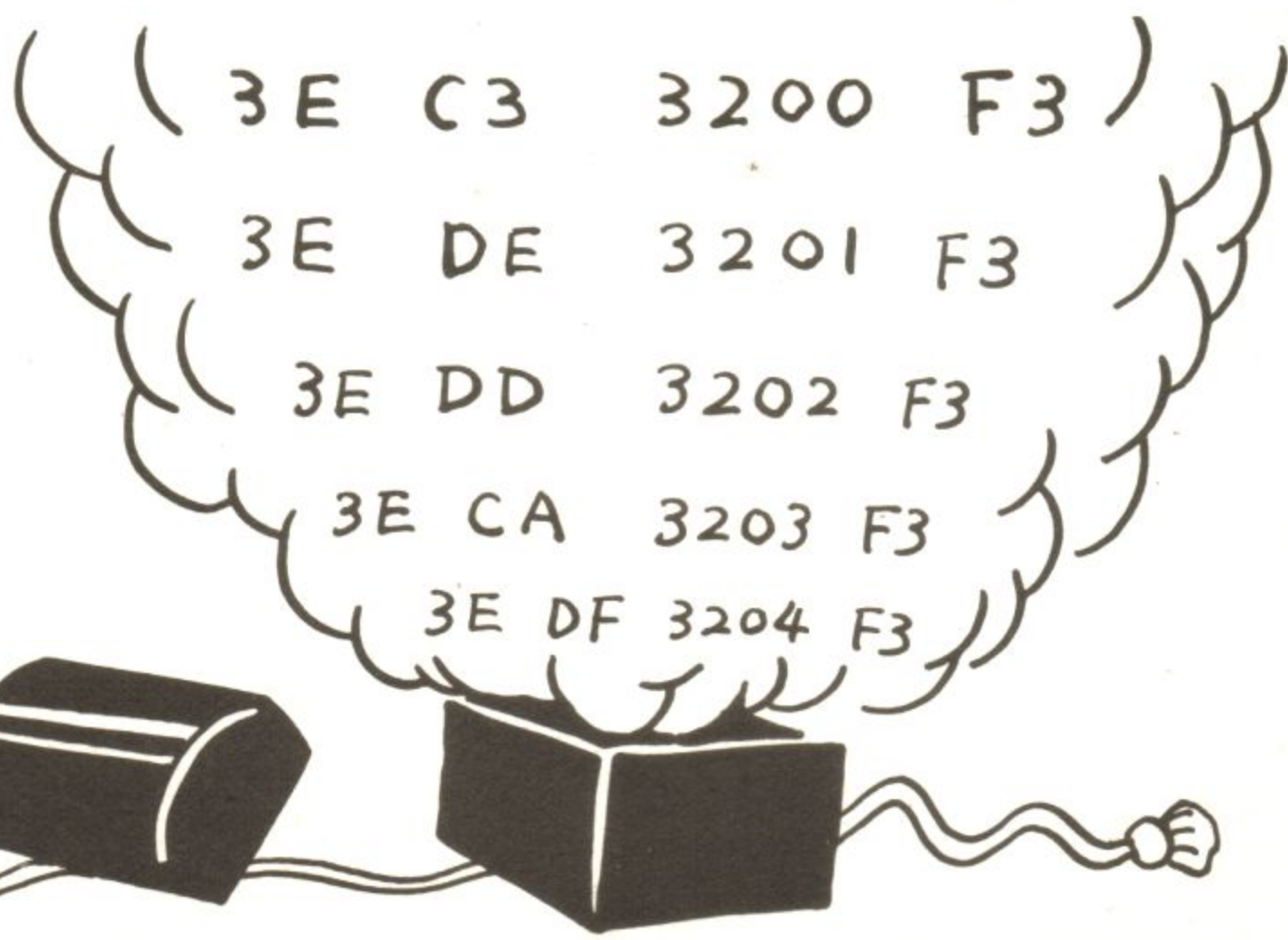
「デンパシンブンシャ」表示

| アドレス | オブジェクト | ニーモニック | コメント |
|--------------|-------------------|----------------------------|---------------|
| D000 D002 | 3E C3 32 00 F3 | LD A, C3H LD (F300H), A | テ |
| D005 D007 | 3E DE 32 02 F3 | LD A, DEH LD (F302H), A | ゝ |
| D00A D00C | 3E DD 32 04 F3 | LD A, DDH LD (F304H), A | ン |
| D00F D011 | 3E CA 32 06 F3 | LD A, CAH LD (F306H), A | ハ |
| D014 D016 | 3E DF 32 08 F3 | LD A, DFH LD (F308H), A | 。 |
| D019 D01B | 3E BC 32 0A F3 | LD A, BCH LD (F30AH), A | シ |
| D01E D020 | 3E DD 32 0C F3 | LD A, DDH LD (F30CH), A | ン |
| D023 D025 | 3E CC 32 0E F3 | LD A, CCH LD (F30EH), A | フ |
| D028 D02A | 3E DE 32 10 F3 | LD A, DEH LD (F310H), A | ゝ |
| D02D D02F | 3E DD 32 12 F3 | LD A, DDH LD (F312H), A | ン |
| D032 D034 | 3E BC 32 14 F3 | LD A, BCH LD (F314H), A | シ |
| D037 D039 | 3E AC 32 16 F3 | LD A, ACH LD (F316H), A | ヤ |
| D03C | C3 66 5C | JP 5C66H | モニタへ ジャンプ! |

その時の参考にと思い、ニーモニック \longleftrightarrow マシン語の対応表より8ビット・ロード命令の項（第25図）をあげておきますので、自分でロード命令を使ってみてください。

次章では、16ビットの数値を一度に転送するための16ビット・ロード命令を紹介します。そろそろ皆さんもマシン語がどんなものかが分かりかけて来たのではないかと思います。

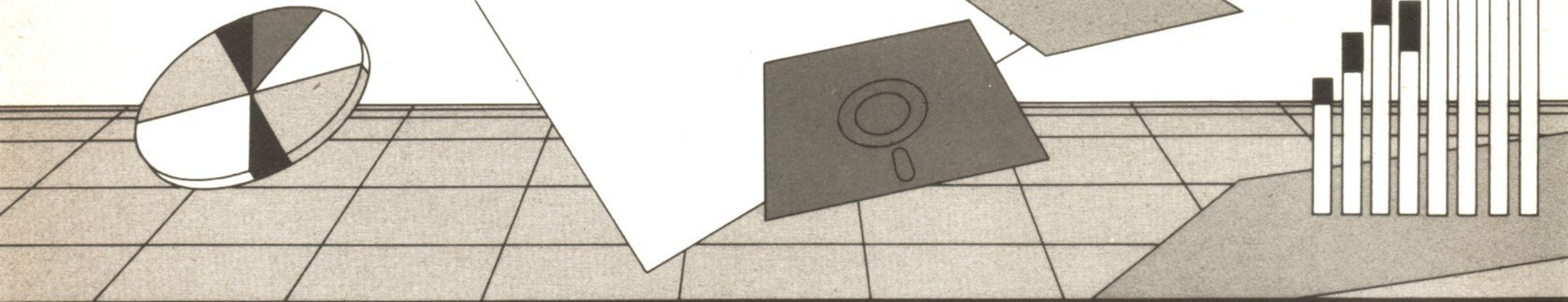
いじめられていたパソコンを
助けて
みると!?



《第25図》 μ COM-82ニーモニック \longleftrightarrow 機械語対照表(8ビット・ロード命令)

| \times | I | R | A | B | C | D | E | H | L | (HL) | (BC) | (DE) | (IX+d) | (IY+d) | (nn) | n |
|--------------------|-----------|-----------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|------|------|------|----------------|----------------|---------------|---------------------|
| LD A, \times | ED 5 7 | ED 5 F | 7 F | 7 8 | 7 9 | 7 A | 7 B | 7 C | 7 D | 7 E | 0 A | 1 A | DD 7 E d | FD 7 E d | 3 A n n | 3 E n |
| LD B, \times | | | 4 7 | 4 0 | 4 1 | 4 2 | 4 3 | 4 4 | 4 5 | 4 6 | | | DD 4 6 d | FD 4 6 d | | 0 6 n |
| LD C, \times | | | 4 F | 4 8 | 4 9 | 4 A | 4 B | 4 C | 4 D | 4 E | | | DD 4 E d | FD 4 E d | | 0 E n |
| LD D, \times | | | 5 7 | 5 0 | 5 1 | 5 2 | 5 3 | 5 4 | 5 5 | 5 6 | | | DD 5 6 d | FD 5 6 d | | 1 6 n |
| LD E, \times | | | 5 F | 5 8 | 5 9 | 5 A | 5 B | 5 C | 5 D | 5 E | | | DD 5 E d | FD 5 E d | | 1 E n |
| LD H, \times | | | 6 7 | 6 0 | 6 1 | 6 2 | 6 3 | 6 4 | 6 5 | 6 6 | | | DD 6 6 d | FD 6 6 d | | 2 6 n |
| LD L, \times | | | 6 F | 6 8 | 6 9 | 6 A | 6 B | 6 C | 6 D | 6 E | | | DD 6 E d | FD 6 E d | | 2 E n |
| LD(HL), \times | | | 7 7 | 7 0 | 7 1 | 7 2 | 7 3 | 7 4 | 7 5 | | | | | | | 3 6 n |
| LD(BC), \times | | | 0 2 | | | | | | | | | | | | | |
| LD(DE), \times | | | 1 2 | | | | | | | | | | | | | |
| LD(IX+d), \times | | | DD 7 7 d | DD 7 0 d | DD 7 1 d | DD 7 2 d | DD 7 3 d | DD 7 4 d | DD 7 5 d | | | | | | | DD 3 6 d n |
| LD(IY+d), \times | | | FD 7 7 d | FD 7 0 d | FD 7 1 d | FD 7 2 d | FD 7 3 d | FD 7 4 d | FD 7 5 d | | | | | | | FD 3 6 d n |
| LD(nn), \times | | | 3 2 n n | | | | | | | | | | | | | |
| LD I, \times | | | ED 4 7 | | | | | | | | | | | | | |
| LD R, \times | | | ED 4 F | | | | | | | | | | | | | |

16ビット・ロード命令 I



はじめに

8ビット・ロード命令はいかがだったでしょうか？
 そろそろ、マシン語の単調さにあきが来てやめ
 たくなった皆さんもいらっしゃるかもしれませんが、もう少し待ってください。確かにマシン語の
 命令一つ一つを取ってBASICのものと比べれ
 ば、はるかに単純でおもしろ味の無いものかもし
 れませんが、その単純な命令を組み合わせでいか
 に複雑な働きをさせるかがマシン語の醍醐味でも
 あるのです。PASCALもFORTRANもCOBOLも
 全てマシン語で動いているのです！

第6章の概要

本章では16ビット・ロード命令を説明します。
 主な働きは8ビット・ロード命令と同じですから
 それほど考えなくても理解できると思います。

後半では、スタックを使ったレジスタ退避命令
 を説明しますがこれもスタックとレジスタ（対）
 との間のロード命令と考える事ができますので本
 章の内容の中に加えさせていただきました。

16ビット・ロード命令

◎16ビットのレジスタ(対)に

直接数値を代入する命令

この命令は、16ビットのレジスタ（対）に16ビ
 ットの数値を代入するだけの働きをするものです
 が、マシン語を使ってプログラムを組んで行く上
 では、最も多く使用する命令の一つだと思います。

本書の冒頭でも登場した

LD HL, F300H

も同じ種類の命令で、HLレジスタ対に16進数の
 F300Hを代入しています。この時当然の事ですが
 HレジスタにはF3Hが、Lレジスタには00H
 がそれぞれ代入されます。

同じようにして、BC, DE, HL, SP, I
 X, IY, の各レジスタ（対）に直接数値を代入
 することができます。

また、この命令をアセンブルすると3～4バイト
 のオブジェクト・コードとなりその中の後2バイト
 には直接代入したい16ビットの数値の上位8ビッ
 トと下位8ビットを逆転したものを割り当てます。

説明だけでは解りにくいと思いますが実際にい
 くつかの命令をアセンブルしてみればすぐわかる
 でしょう。

まず先程の命令をE000Hからのマシン語にアセ

ンブルすると、

アドレス オブジェクト (16進)

E 0 0 0 21 00 F 3

となります。この場合始めの“21”は、HLレジスタ対に16ビットの数値を代入するという命令の部分で次からの“00 F3”で、代入する16進数を与えています。

例えば、IXレジスタに16進数の5678Hを代入したければ、

LD IX, 5678H

という命令を使いますが、この命令をアセンブルすると、“DD 21 78 56”の4バイト命令になりやはり、うしろ2バイトが代入したい数値を与えています。

8ビット・ロード命令時に説明した、

LD A, ECH

などを16ビットの命令に直したものと考えれば良いでしょう。

◎16ビットのレジスタ(対)と特定の

2バイトに渡るメモリ間のロード命令

この命令がロード命令の中でも比較的わかりにくい命令だと思います。何度もくり返して説明して来たように、Z80を含めた80系のCPUは、2バイトに渡る数値をメモリ上に置く場合は必ず上位、下位を逆にして下位バイトの方をメモリの若い方に入れます。

この方式に慣れてしまえば何ということはないのですが、それでも2バイトの数を扱う場合に少し考え込んでしまうこともあります。

それでは実際の使用例を考えてみましょう。例えば現在何らかのゲーム・プログラムを組んでいて、E000H～E001H番地の2バイトに渡ってスコアが入っているものとします。ゲームが1回終了した時点でハイ・スコアを得た場合には今度はこのスコアをハイ・スコアとして登録しなければなりません、ハイ・スコアはE002H～E003H番地に入れるものとします。

こんな場合はロード命令を使ってE000H～E001H番地の内容をE002H～E003H番地に移さなければなりません、まずこの問題を前章で説明した8ビット・ロード命令のみを使ってプログラムを組むと次のようになります。

| | | |
|--------------------|---|----------------------|
| LD L, (E000H) | } | スコアをHLレジスタ対に入れる |
| LD H, (E001H) | | |
| LD (E002H), L | } | HLレジスタ対の内容をハイ・スコアに登録 |
| LD (E003H), H | | |

このようにして、上位8ビットと下位8ビットとを一度に転送できず二度に分けなければならないのは何かと不便ですので16ビットのロード命令が必要になるのです。上記のと同じプログラムを16ビット・ロード命令を使って組むと

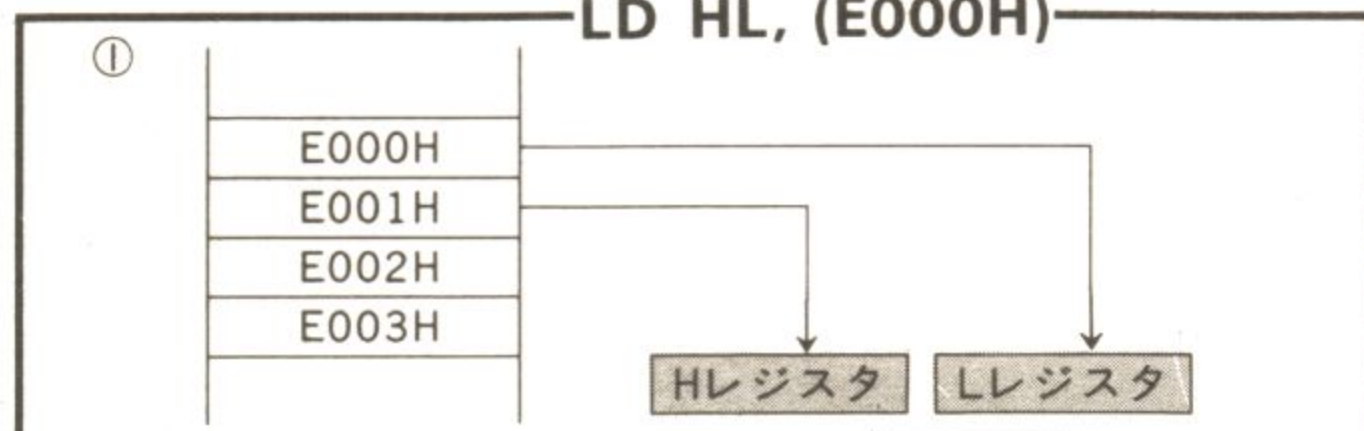
LD HL, (E000H)
LD (E002H), HL

となります(第26図参照)。

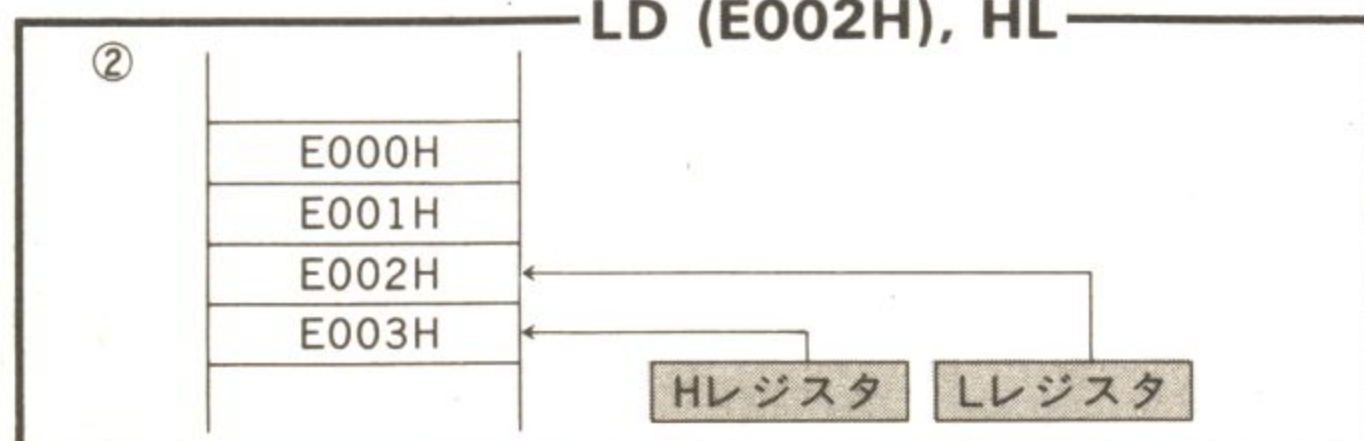
前のプログラムと比較すると使用するメモリが半分になりますし、また4ステップの命令と2ステップにまとめた命令では後で見た時の見易さが全然ちがうのではないのでしょうか。

このように16ビット・ロード命令はプログラム

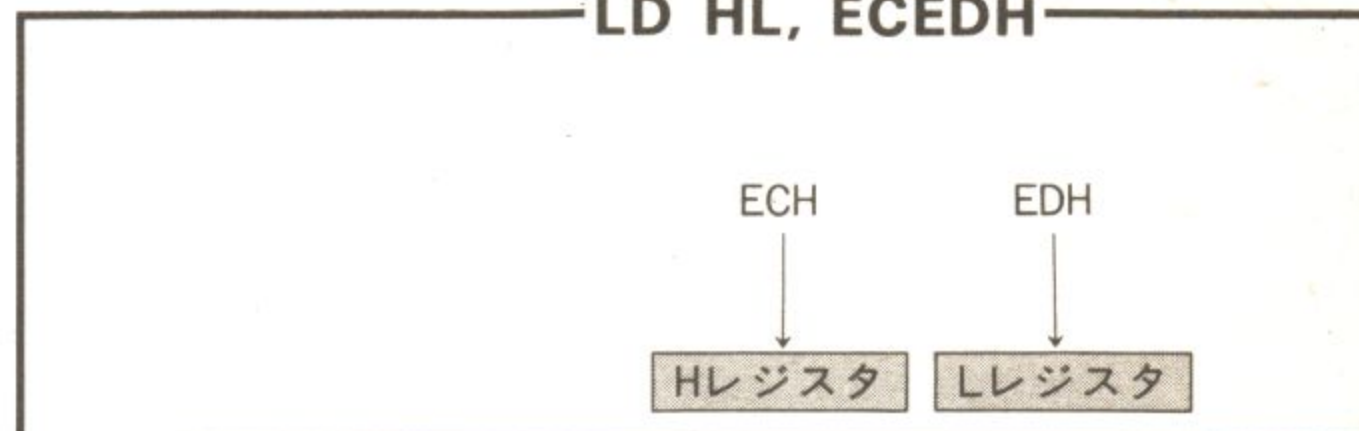
《第26図》16ビット・ロード命令 I
LD HL, (E000H)



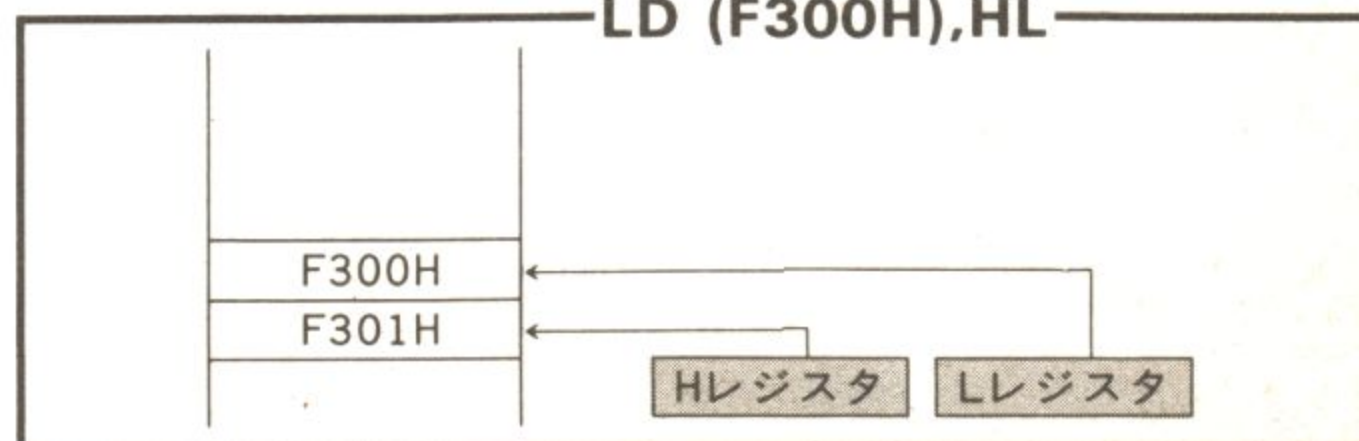
LD (E002H), HL



《第27図》16ビット・ロード命令 II
LD HL, ECDH



LD (F300H), HL



を見易く、組み易くするためには不可欠なものなのです。

ここでもう一度この命令について例をあげて説明します。

例えば、80字モードの時に

```
LD HL, ECEDH
```

```
LD (F300H), HL
```

を実行するとどうなるでしょうか(第27図参照)?

まず始めの命令によって、HLレジスタ対に16進数のECEDHが入りますが、この時HレジスタにはECHが、LレジスタにはEDHが入ります。

次の命令で、F300H～F301H番地にHLレジスタ対の内容である、ECEDHが転送されるのですが、下位バイトがアドレスの若い方にロードされますので、F300H番地にはEDHが、F301H番地にはECHが、それぞれ送られます。その結果テレビ画面の“LOCATE 0, 0”の位置には“○”のキャラクタが“LOCATE 1, 0”の位置には“●”のキャラクタが表示されます。

もちろん実際にPC—8001で実行してみる場合には、最後に

```
HALT
```

命令を入れて実行を止めるか、

```
JP 5C66H
```

によってモニタにジャンプさせるかしなければなりません。

また、逆に

```
LD HL, (F300H)
```

を行う事によって、F300H～F301H番地にどんなキャラクタ・コードが入っているのかをHLレジスタ対にもって来ることができます。

以上でこの命令の使用方法が大体わかっていただけたと思いますが、この命令も3～4バイトのオブジェクトにアセンブルされ、うしろ2バイトがアドレスを指定します。また、本章のプログラム例では16ビットのレジスタ(対)としてHLレジスタ対を使用しましたが、実際は、BC, DE, HL, SP, IX, IYの各レジスタ(対)を使う事ができます。

以上でニーモニックに“LD”が付く命令は全て説明した事になります。8080CPUの頃は、同じロード命令でも“MOV, MVI, LXI”などのニ

ーモニックに分かれていて覚えにくかったものでした。

レジスタの退避

さて、Z80にはいくつかの汎用レジスタが用意されていますが、通常のプログラムでは、ほとんどの汎用レジスタをフルに使用します。全てのレジスタを使っている時、もし少しの間どうしてもレジスタを使用しなくなったらどうすればよいでしょうか? 当然、汎用レジスタの内容は全てもと通りにもどしておかなければなりません。

例えばHLレジスタ対を一時的にあげる場合にロード命令を使って、

```
LD (E000H), HL
```

のようにHLレジスタ対の内容を、E000H～E001H番地に退避しておいてから、何らかの作業によってHLレジスタ対を使用し再度ロード命令によって、

```
LD HL, (E000H)
```

のようにHLレジスタ対の内容をもとにもどしておけばよいのです。

しかしこの方法でレジスタ(対)の内容を退避しておくのには、いくつかの問題点があります。

まず上記の方法ですとレジスタの退避先アドレスをいちいち指定しなくてはいけませんからレジスタ(対)の退避先アドレスを全て記憶しておかなければ、後で正しいレジスタ(対)に値をもどすことができません。下手をすると同じアドレスに2度退避しておいて気付かず、後でデバッグの時に大へんな苦勞をすることも有り得ます。

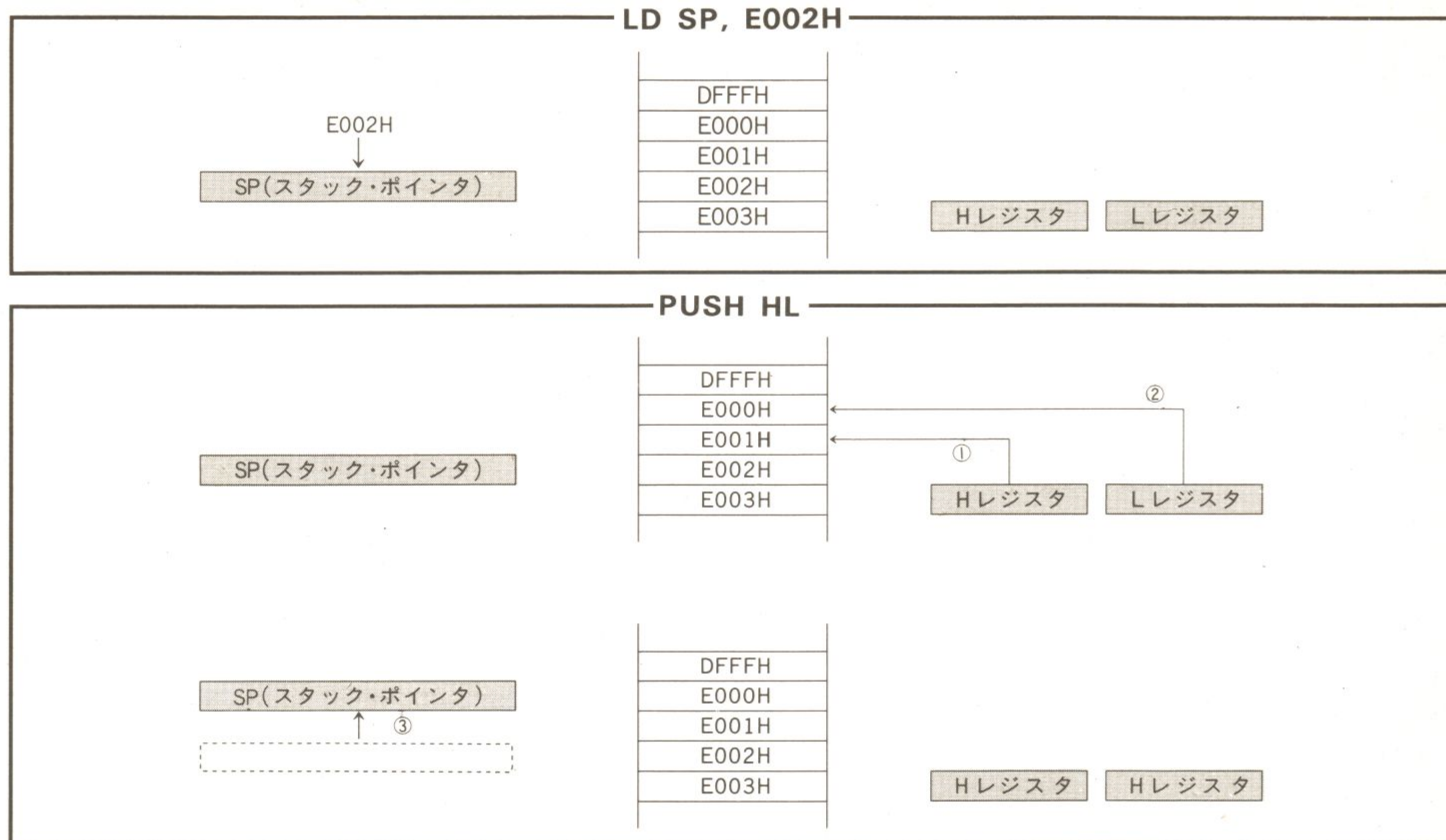
又、このロード命令では必ず3～4バイトずつ使いますからメモリの無駄使いにもなります。

そこでメモリへのレジスタ退避をより効率よく行うためにスタックの考え方が必要になるのです。

スタックとは……

それでは、スタックとはいったい何なのでしょううか?

《第28図》レジスタ退避命令 (PUSH命令)



マシン語やアセンブリ言語などと言うスタックとは単にメモリの使い方なのです。先程の例でもわかる様に普通のロード命令で、レジスタ(対)←メモリ間の転送を考えれば、そのたび事にメモリのアドレスを指定してやらなければなりません。

そこでメモリのアドレスを指定するためのレジスタを、専用に使いそれを自動的に増減してやればレジスタ(対)←メモリ間の転送のたびにアドレスを指定してやる必要がなくなるわけです。

皆さんも、もうお気づきの事と思いますが、このアドレス指定専用のレジスタが、SP (スタック・ポインタ) なのです。そしてメモリ内の、SP (スタック・ポインタ) によって指定されるエリアをスタックと呼び、レジスタ(対)←メモリへの転送をレジスタ退避、逆にメモリ→レジスタ(対)の転送をレジスタ退避解除と言います。

レジスタ退避(解除)命令

◎レジスタ退避命令(PUSH命令)

実際にレジスタ退避命令を使ってHLレジスタ

対の内容をメモリ上のスタック・エリアに退避する場合を考えてみましょう (第28図参照)。

LD SP, E002H

PUSH HL

まず、ロード命令でSP (スタック・ポインタ) にE002Hを代入しています。次にPUSH命令によってHLレジスタ対の内容をスタックにPUSH (押し込み) しているのですが、PUSH命令の動作を分解すると以下ようになります。

- ①, SP—1番地にHレジスタの内容をロード
- ②, SP—2番地にLレジスタの内容をロード
- ③, SPの内容から2を減じておく

では、この動作を追ってみましょう。

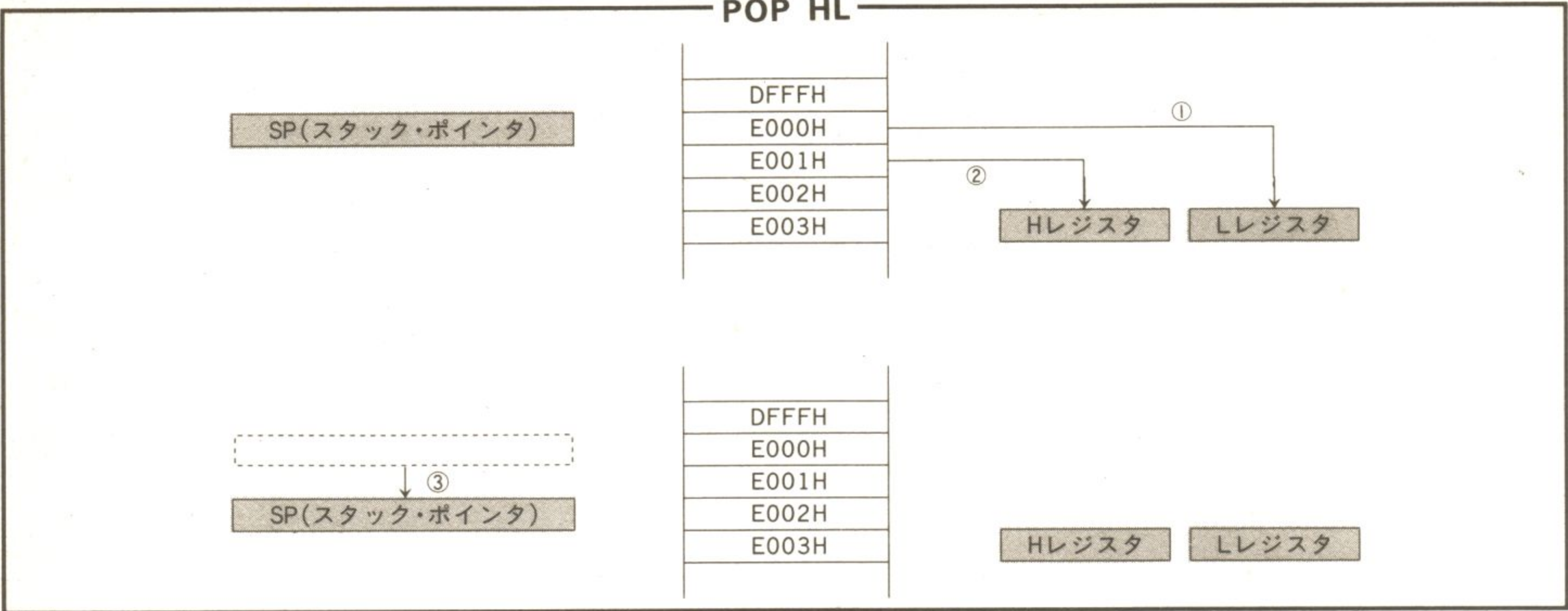
①によって、E002H—1番地つまりE001H番地にHレジスタの内容がロード (退避) されます。

②で、E002H—2番地つまりE000H番地にLレジスタの内容がロード (退避) されます。

③で、今までE002HだったSPがE000Hになります。

結果として、E000H～E001H番地にHLレジスタ対の内容が退避された事になり、先程の

《第29図》レジスタ退避解除命令（POP命令）



《第30図》16ビット・ロード命令(ニーモック↔機械語)対応表

| × | AF | BC | DE | HL | SP | IX | IY | nn | (nn) |
|------------|-----|------------------|------------------|------------|------------------|------------------|------------------|------------------|------------------|
| LD AF, × | | | | | | | | | |
| LD BC, × | | | | | | | | 0 1 n n | ED 4 B n n |
| LD DE, × | | | | | | | | 1 1 n n | ED 5 B n n |
| LD HL, × | | | | | | | | 2 1 n n | 2 A n n |
| LD SP, × | | | | F 9 | | DD F 9 | FD F 9 | 3 1 n n | ED 7 B n n |
| LD IX, × | | | | | | | | DD 2 1 n n | DD 2 A n n |
| LD IY, × | | | | | | | | FD 2 1 n n | FD 2 A n n |
| LD (nn), × | | ED 4 3 n n | ED 5 3 n n | 2 2 n n | ED 7 3 n n | DD 2 2 n n | FE 2 2 n n | | |
| PUSH × | F 5 | C 5 | D 5 | E 5 | | DD E 5 | FD E 5 | | |
| POP × | F 1 | C 1 | D 1 | E 1 | | DD E 1 | FD E 1 | | |

LD (E000H), HL

を行ったのと同じ事になります。しかも、SP (スタック・ポインタ) の内容はPUSH命令のたびに2ずつ引かれて行きますから、PUSH命令を連続して使う事によりメモリの若い方へ若い方へと次々にレジスタ (対) の内容を退避して行く事が可能となり、それがSP (スタック・ポインタ) の役割でもあるのです。

◎レジスタ退避解除命令(POP命令)

今度はスタック上に退避したデータをレジスタ (対) にもどす場合を考えてみます (第29図参照)。

POP HL

このPOP命令によってスタック上のデータがPOP (引き出し) されるのですが、POP命令の動作を分解すると以下のようになります。

- ①. LレジスタにSP番地の内容をロード
- ②. HレジスタにSP+1番地の内容をロード
- ③. SPの内容に2を加える

先程PUSHした内容をHLレジスタ対にPOPでもどす場合はどうなるでしょうか?

①によって、SP (スタック・ポインタ) の指すアドレスの内容がLレジスタにロードされますが、この場合、SP (スタック・ポインタ) の内容はE000Hになっていますから、E000H番地の内容がLレジスタに入ります。

②で、E000H+1番地つまりE001H番地の内容がHレジスタに入ります。

③で、今までE000HだったSPがE002Hになります。

結果として、E000H~E001H番地の内容がHLレジスタ対にもどされた事になり、

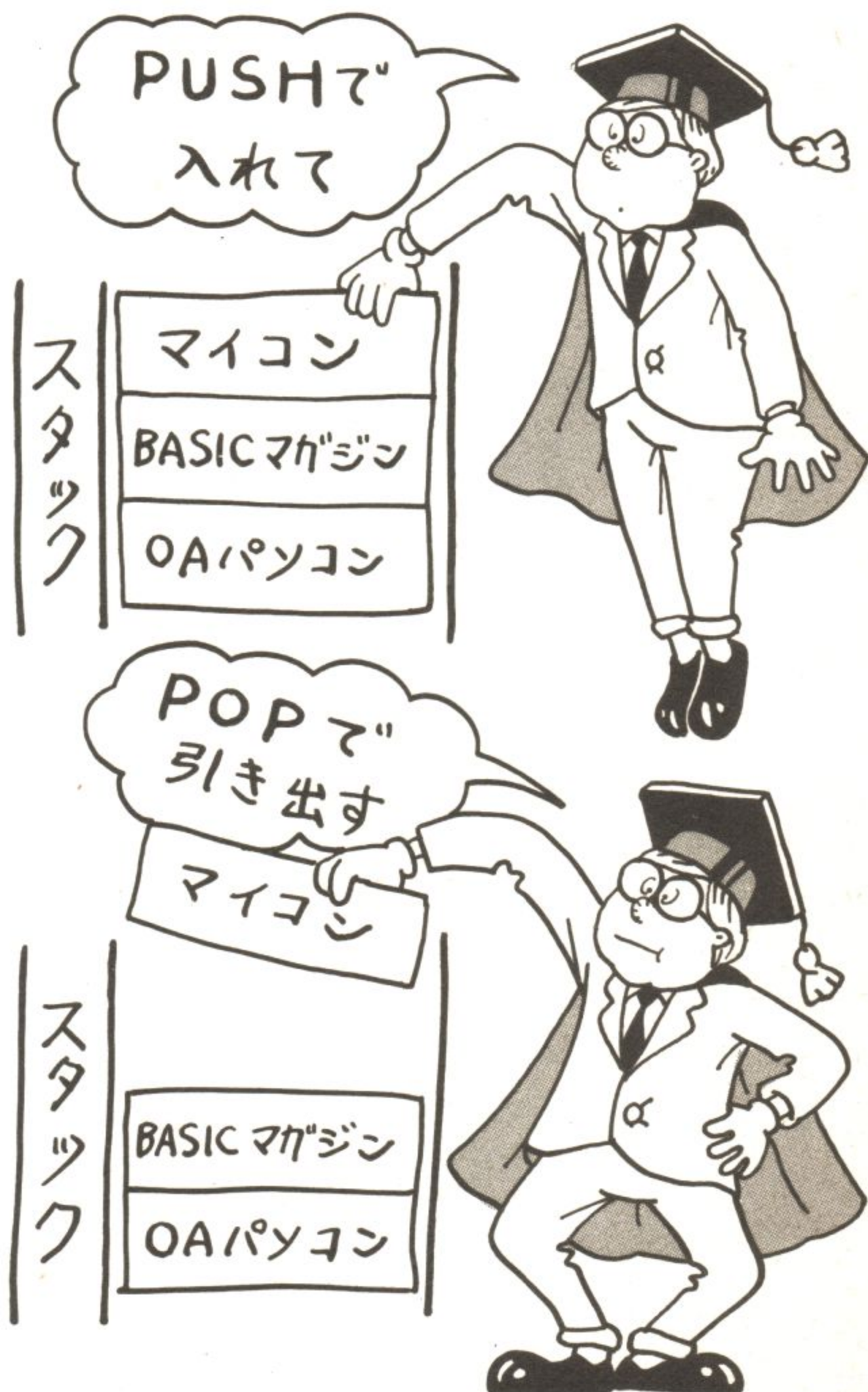
LD HL, (E000H)

を行ったのと、同じ事になります。しかも、SP (スタック・ポインタ) はPOP命令のたびに2ずつ加算されていきますから、POP命令を連続して使う事により、メモリの若い方から次々にレジスタ (対) に値をもどして行く事が可能となります。結局POP命令は、PUSH命令の全く逆の命令なのです。

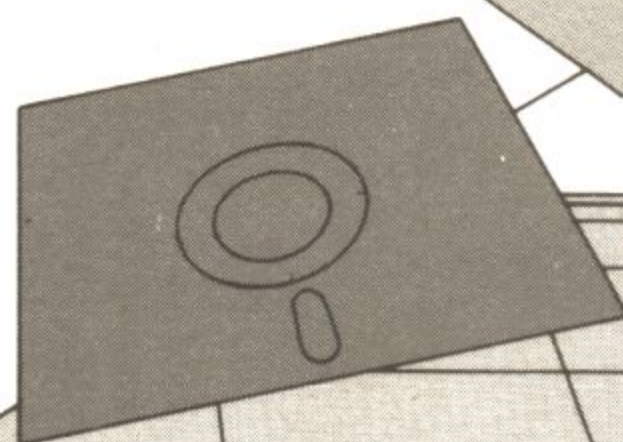
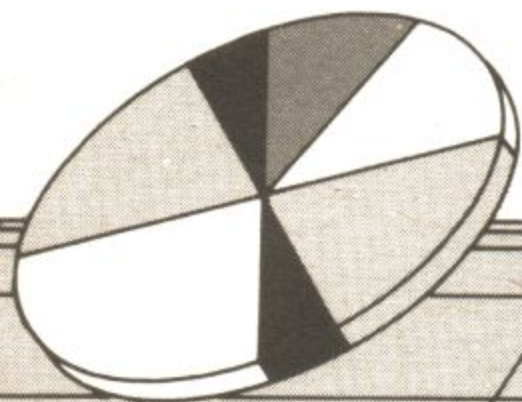
レジスタ退避(解除)命令のまとめ

以上、PUSH命令、POP命令共にHLレジスタ対を例にとって説明しましたが、十分に御理解いただけたでしょうか? なおこの命令は、AF, BC, DE, HL, IX, IYの各レジスタ (対) に対して有効ですから各レジスタ (対) の内容をスタックにPUSH, POPすることができます。また、1~2バイトのオブジェクトとなりますので省メモリのためにも非常に重要な命令といえましょう。なお、この命令を使ってレジスタ (対) の内容を一時退避する場合には、SP (スタック・ポインタ) に入っている値を常に注意する必要があります。

ちなみに、N-BASICからマシン語のサブルーチンをコールする場合には14バイト (7レベル) のスタックが確保されています。



16ビット・ロード命令II



前章では、Z80におけるスタックの働きを説明しましたがどんな感想を持たれたでしょうか？

本章では、このスタックへのレジスタ退避を使った応用例と実際にこの命令を使う現場での使用方法を示して、スタックへのレジスタ退避(解除)の項をしめくくり、今度はレジスタ(対)の内容を他のレジスタ(対)やスタックと交換するエクスチェンジ命令を紹介します。

今まで登場した命令全てを使ったプログラム例も紹介しますのでお楽しみに！

レジスタ退避(解除)命令応用例

レジスタ退避(解除)命令についての応用方法を少し考えてみましょう。

まず、16ビットのレジスタ(対)間でロード命令を実行する場合、例えばBCレジスタ対の内容をDEレジスタ対に移す場合、8ビットのロード命令を使って、

```
LD    D, B
LD,   E, C
```

の様に行う場合もありますし、一度メモリ上を介して16ビットのロード命令を使う方法もありますが、少しおもしろい方法として、

```
PUSH  BC
POP   DE
```

を行う場合も考えられます。まずBCレジスタ対

の内容をスタックへ退避しておいて、今度はスタックの内容をDEレジスタ対に引き出せば結果としてBCレジスタ対の内容をDEレジスタ対に送ったのと同じ事になるのです。

特に、IXやIYなどのインデックス・レジスタでは、8ビットのロード命令を使う事もできませんからどうしてもメモリ上を介する事になります。IXレジスタからHLレジスタ対に内容を転送する場合を考えて例をあげてみましょう。まず、16ビットのロード命令を使って、E100H~E101H番地を介する場合は、

```
LD    (E100H), IX
LD    HL, (E100H)
```

となります。これをスタックを介するようにかえると

```
PUSH  IX
POP   HL
```

になります。アセンブルすると前者が7バイトかかるのところ、後者は3バイトですみますから、後者の省メモリ性がわかると思います。

次に、この命令を使って二つのレジスタ(対)の内容を交換してみましょう。例えば、IXレジスタとIYレジスタの内容を交換する場合は、

```
PUSH  IX
PUSH  IY
POP   IX
POP   IY
```

となります。PUSH命令では最後にPUSHし

た内容からしかPOPすることができませんから最初の

```
POP    IX
```

で、すぐ前にPUSHしたIYレジスタの内容をIXレジスタに移し、次の

```
POP    IY
```

で、一番初めにPUSHしたIXレジスタの内容をIYレジスタに移しています。

始めのうちはわかりづらいと思いますが、秘伝を教えましょう。

```

PUSH    IX
PUSH    IY
-----
POP     IX
POP     IY

```

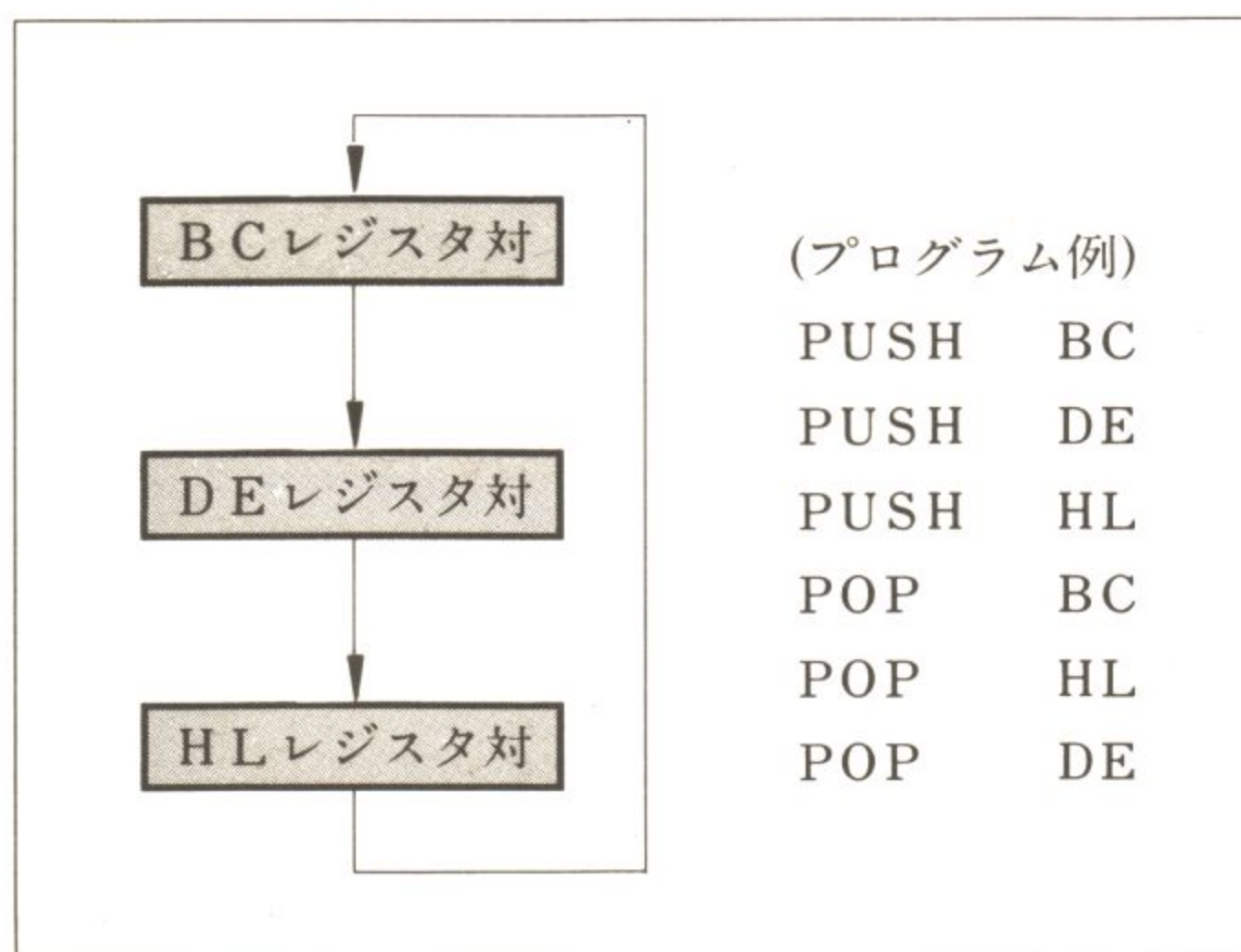
IXからIYへ
IYからIXへ

このように、PUSH命令とPOP命令の間で区切りその区切りを境に対称に考えればすぐにどのレジスタ(対)からどのレジスタ(対)に値を転送しているかがわかるのです。これは、レジスタ(対)が3組以上になった場合でも、また、同じレジスタ対に値をもどす場合でも同様です。

今度は3組のレジスタ(対)で考えてみましょう。第31図のように、BC, DE, HLの各レジスタ対の内容を一つずつずらして行く方法を考えてみてください。

普通に考える場合は慣れないとたいへんですが、先程の秘伝を使って図を書けばすぐにわかりますね。解答は、

《第31図》 3組のレジスタ(対)の交換



```

PUSH    BC
PUSH    DE
PUSH    HL
POP     BC
POP     HL
POP     DE

```

BCからDEへ
DEからHLへ
HLからBCへ

となります。同じ事を行うのに、

```

PUSH    DE
PUSH    BC
PUSH    HL
POP     BC
POP     DE
POP     HL

```

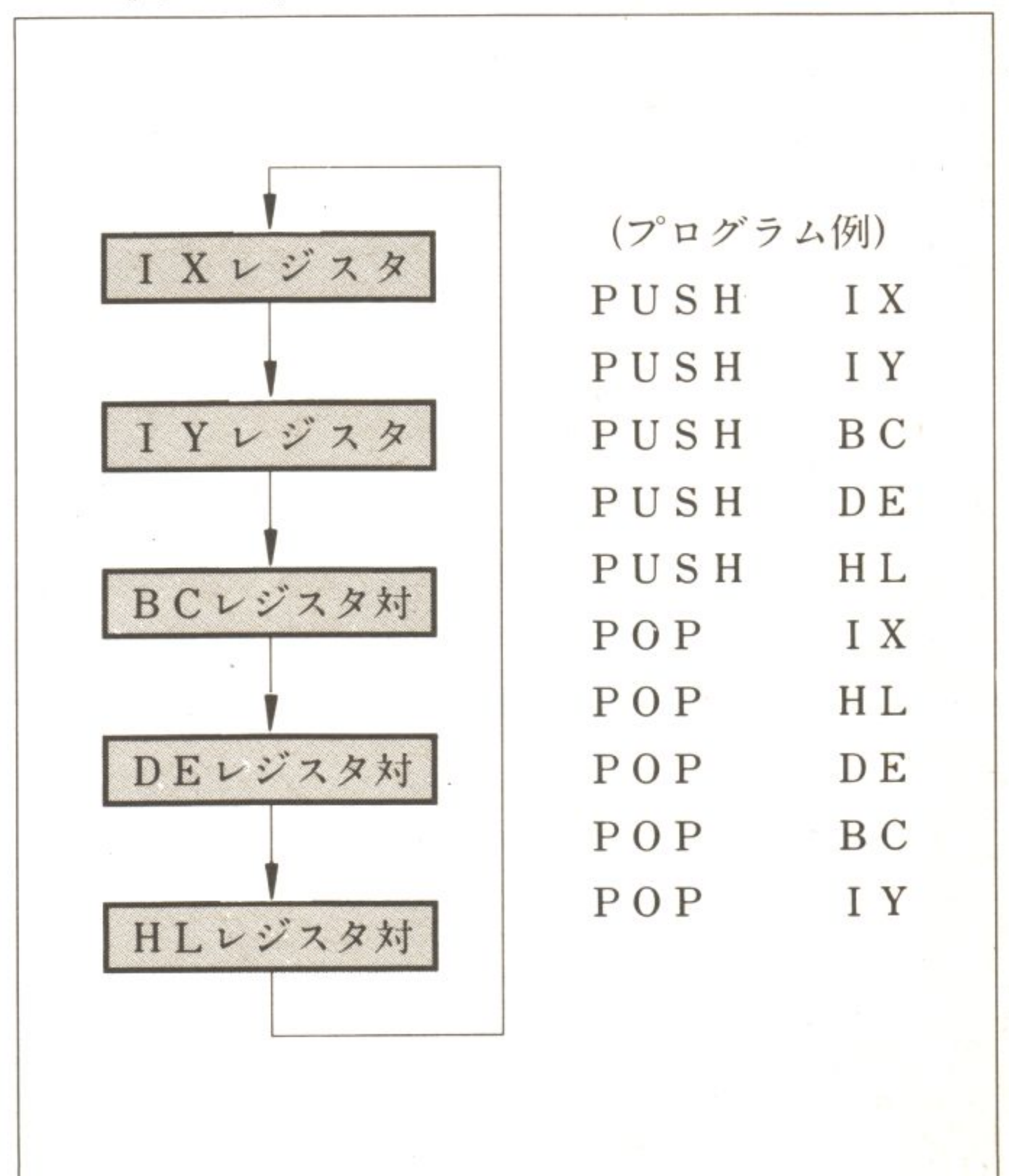
DEからHLへ
BCからDEへ
HLからBCへ

などいくつかの方法がありますがどの方法を行っても結果は同じです。

次にレジスタ(対)が5組の場合の例を第32図にあげておきます。興味のある方は御自分で命令を追ってみてください。

スタックを使った応用例は他にもいくつかあります。知っていればすぐ役立つというものでもありませんが一般のプログラムなどにも確実に使われている使用法なので覚えておいてください。

《第32図》 5組のレジスタ(対)の場合



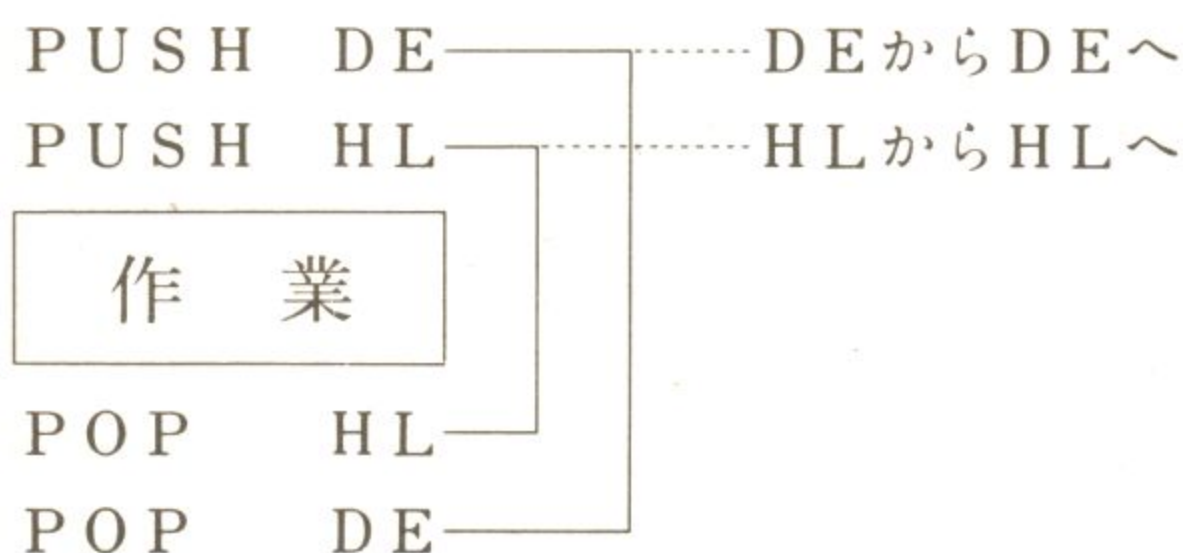
レジスタ退避(解除)命令使用例

読者の皆さんもスタックの考え方にかなり親しみを持ってきたと思いますが、ここでもう一度この命令の使い方について、実際の使用例を中心に考えてみましょう。

前にも少し説明したように、この命令は、レジスタ(対)の内容をスタックに退避するためPUSH命令と、スタックの内容をレジスタ(対)にもどすPOP命令に分けることができます。

PUSH命令とPOP命令が連続して使われる事はほとんどなく、また、応用例の所で述べた様な使い方よりも、もともと内容の入っていたレジスタ(対)に値をもどす様な使い方が一般的です(第33図)。

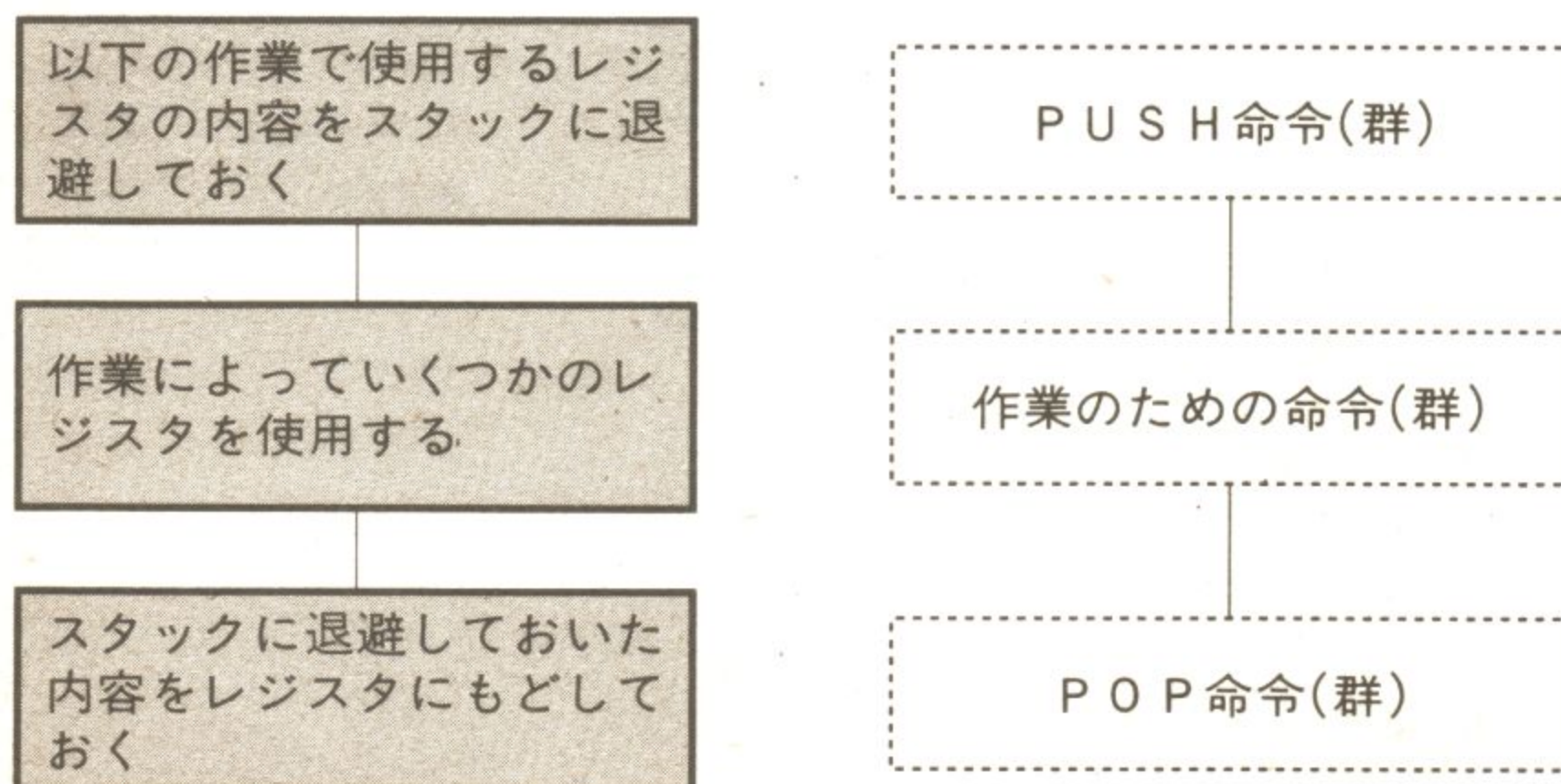
ですから、DE、HLの各レジスタ対を何かの作業のために退避しなければならない場合には、普通以下の様な使い方をします。



当然、退避するレジスタ(対)が1組の場合でも3組以上の場合でも同様な使い方をします。

一見あまり使用することのない命令のように見えるかもしれませんが、レジスタ(対)の内容を退避しておきたい事はかなり多くありますので、

《第33図》PUSH命令・POP命令の一般的な使われ方



実際のプログラム、たとえばN-BASICのROMなどを逆アセンブルすれば、この命令が多く使われているのにおどろくほどです。

エクスチェンジ命令

◎16ビットのレジスタ(対)

の内容を交換する命令

例えば、DEレジスタ対とHLレジスタ対の内容を交換したい時はどうすればよいでしょうか？

まず第1の方法として先程のレジスタ退避を使って

```

PUSH DE
PUSH HL
POP  DE
POP  HL
    
```

の4ステップを実行すれば両レジスタ対の内容は見事に交換されます。

第2に、Aレジスタでも余っていれば、

```

LD  A, E }
LD  E, L } E↔Lレジスタを交換
LD  L, A }

LD  A, D }
LD  D, H } D↔Hレジスタを交換
LD  H, A }
    
```

を実行すれば、ロード命令だけで交換する事ができます。

他にも一度メモリ上を介する方法などいくつか

《第34図》エクスチェンジ命令

| | |
|-------------|-----------|
| EX AF, AF' | 0 8 |
| EX DE, HL | E B |
| EX (SP), HL | E 3 |
| EX (SP), IX | DD E 3 |
| EX (SP), IY | FD E 3 |
| EXX | D 9 |

の方法が考えられますが、いずれも数ステップ数バイトの命令になってしまいます。

そこでエクスチェンジ命令を使うわけです。エクスチェンジ命令はその名のとおり内容を交換する命令です。上記のDEレジスタ対とHLレジスタ対の内容を交換する場合はエクスチェンジ命令を使って、

EX DE, HL

の一つの命令で行う事ができます。しかしこの命令は16ビットのレジスタ(対)全てに対して有効なわけではなく、今の

EX DE, HL

の他には、表レジスタのAFと裏レジスタのAF'を交換する

EX AF, AF'

と、表レジスタのBC, DE, HLと裏レジスタのBC', DE', HL'の3組のレジスタ対の内容を同時に交換する

EXX

の合計3個のエクスチェンジ命令が使えるのみです。

◎ 16ビットのレジスタ(対)と

スタックの間で内容を交換する命令

この命令は16ビットのレジスタ(対)と2バイトに渡るメモリの間で値を交換する命令ですが、レジスタ退避のためのPUSH命令、POP命令などと同じ様にメモリ上のアドレスを指定するのに、SP(スタック・ポインタ)を使用します。

具体的には、POP命令を行った場合にレジス

タ(対)に入るべき値がレジスタ(対)に代入されると同時に、現在までその値が入っていたアドレスにレジスタ(対)の内容がPUSHされたのと同じ事になります。

説明では、わかりにくいと思いますので、一つ例をあげてみましょう。

実際にはこの命令は、

EX (SP), HL

のようなニーモニックで表わしますが、この命令を以前説明を終えた、他の命令に置き換えてみると

POP DE

EX DE, HL

PUSH DE

の3ステップになってしまいます。しかも、HLレジスタ対の内容と、スタックの内容を交換するために全然関係ないレジスタ対、この場合はDEレジスタ対をどうしても仲介のためだけに使用しなければなりません。

このように一見して、あまり目立たないエクスチェンジ命令でも、無ければ大へんむだな事をしなければならぬ事がわかると思います。

また、HLレジスタ対を例にとって説明しましたが、実際はIX, IYの各レジスタとスタック(メモリ)の間で内容を交換する事もできます。

プログラム例の説明

それでは、今までに説明した、ロード命令、レ

《第35図》 プログラム例

| アドレス | オブジェクト | ニーモニック | コメント |
|------|-------------|---------------|------------------------|
| E000 | 31 10 F3 | LD SP, F310H | SPにF310Hを代入 |
| E003 | 01 E8 E9 | LD BC, E9E8H | BCレジスタ対にE9E8Hを代入 |
| E006 | C5 | PUSH BC | BCレジスタ対の内容をスタックに退避 |
| E007 | 21 EA EB | LD HL, EB EAH | HLレジスタ対にEB EAHを代入 |
| E00A | E3 | EX (SP), HL | スタックの内容とHLレジスタ対の内容を交換 |
| E00B | E5 | PUSH HL | HLレジスタ対の内容とスタックに重ねて退避 |
| E00C | ED 73 00 E1 | LD(E100H), SP | E100H~E101H番地にSPの内容を移す |
| E010 | C3 66 5C | JP 5C66H | マシン語モニタのコマンド待ちへジャンプ |

ジスタ退避命令，エクスチェンジ命令を使ったプログラム例を考えてみましょう。

第35図のプログラムを見てください。このプログラムを見て何をやっているプログラムだかわかりますか？ わからないからといってがっかりしないでください。一見して単純そうに見えるこのプログラムでも実は、今まで説明した命令を全て熟知していなければわからない様になっているのですから。

では、このプログラムを順に説明して行きます。まず

LD SP, F310H

によって、SP（スタック・ポインタ）に、F310Hを代入しています。これでF30FH番地からアドレスの若い方へ向かってスタック領域がとられたことになり、レジスタ退避命令を行った時に各レジスタ（対）の内容がF30FH番地以前に退避されていく事になります。次に

LD BC, E9E8H

によってBCレジスタ対にE9E8Hを代入した後

PUSH BC

でBCレジスタ対の内容をスタックに退避しています。この時、始めにBレジスタの内容であるE9HがF30FH番地に入り、次にCレジスタの内容であるE8HがF30EH番地に入り、最後に今までF310Hが入っていたSP（スタック・ポインタ）の内容が自動的に-2されて、F30EHに変わります。次の

LD HL, EBDAH

でHLレジスタ対にEBDAHを代入した後、エクスチェンジ命令

EX (SP), HL

を使って、SP（スタック・ポインタ）で指定するアドレスから2バイトの内容と、HLレジスタ対の内容とを同時に交換しています。この時SP（スタック・ポインタ）は、F30EH番地を指定してますからF30EH～F30FH番地とHLレジスタ対の間で内容を交換することになり、結果としてF30EH番地にLレジスタの内容であったEAHが、F30FH番地にはHレジスタの内容であったEBHが、そしてHLレジスタ対

には、先程PUSHした。E9E8Hが入ります。その後すぐ

PUSH HL

で、HLレジスタ対の内容である、E9E8Hをスタックに退避しています。この時SP（スタック・ポインタ）には、F30EHが入っていますから、F30DH番地にE9Hが、F30CH番地にE8Hが入りSP（スタック・ポインタ）の内容が-2されて、F30CHに変化します。そして、そのSP（スタック・ポインタ）の変化が事実がどうかを確認するために

LD (E100H), SP

を実行して、E100H～E101H番地にSP（スタック・ポインタ）の内容を入れていますのでこのプログラムを実行し終った後E100H～E101H番地の内容をモニタのD（ダンプ）コマンドで調べてみれば、SP（スタック・ポインタ）の内容であるはずのF30CHが上位8ビットと下位8ビットが逆転されて、E100H番地には0CHが、E101H番地にはF3Hがそれぞれ入っているのが確認できるはずです。そして最後の

JP 5C66H

によって、PC-8001のマシン語モニタのスタート・アドレスである5C66H番地にジャンプします。

以上、順を追って説明しましたが、この結果重要なのは、F30CH番地にはE8Hが、F30DH番地にはE9Hが、F30EH番地にはEAHが、F30FH番地にはEBHがそれぞれ収納されることです。もちろんそれぞれのアドレスはV-RAM上ですから、PC-8001の画面上、最上段には、キャラクタ・コードのE8H～EBHに対応するキャラクタ、つまり“♠♥♦♣”が1列に表示されることになります。ただし、40字モードの場合は一つおきに二つのキャラクタが表示されることになります。

以上、かなり特殊なプログラムですから解説するのもにも大へんな手間だと思いますがぜひもう一度読みなおしてください。そして、このプログラム例が完全に理解できなかった場合には、自分のわからない命令を調べ直して欲しいと思います。

この部分が完全に理解できなかったからといって
も別に何も起こりませんが後にマシン語を勉強し
て行く過程で必ず知りたくなる時が来るはずです。

まとめ

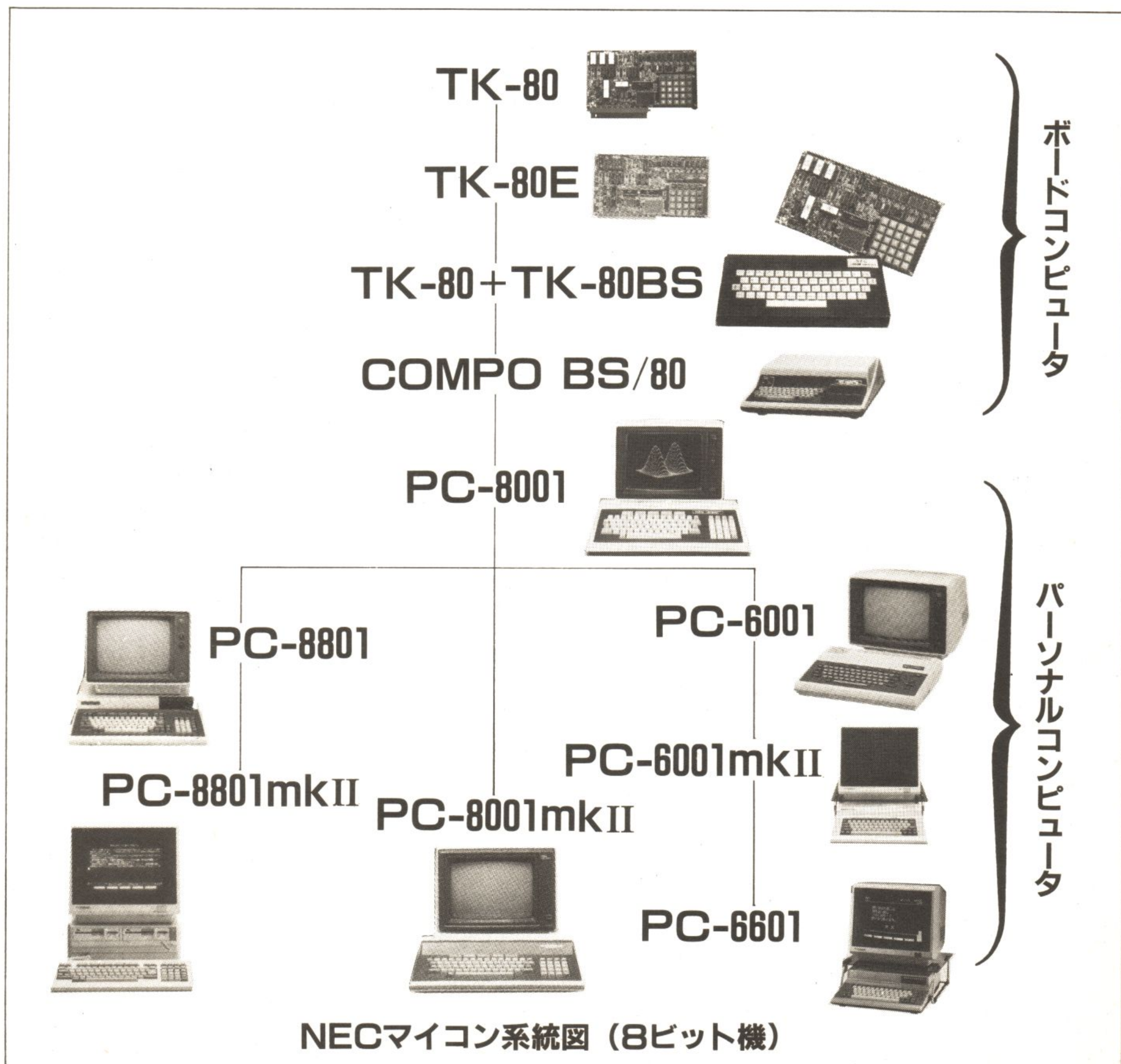
PC-8001を使っている皆さんのほとんどは、
高級言語 (N-BASIC) → マシン語
の経路で、パーソナル・コンピュータの学習を進
めて来られた事と思います。

BASICのように簡単に使える言語が標準装
備されている事はビジネス面への応用を考えると
大へんすばらしい事なのですが、BASICの命

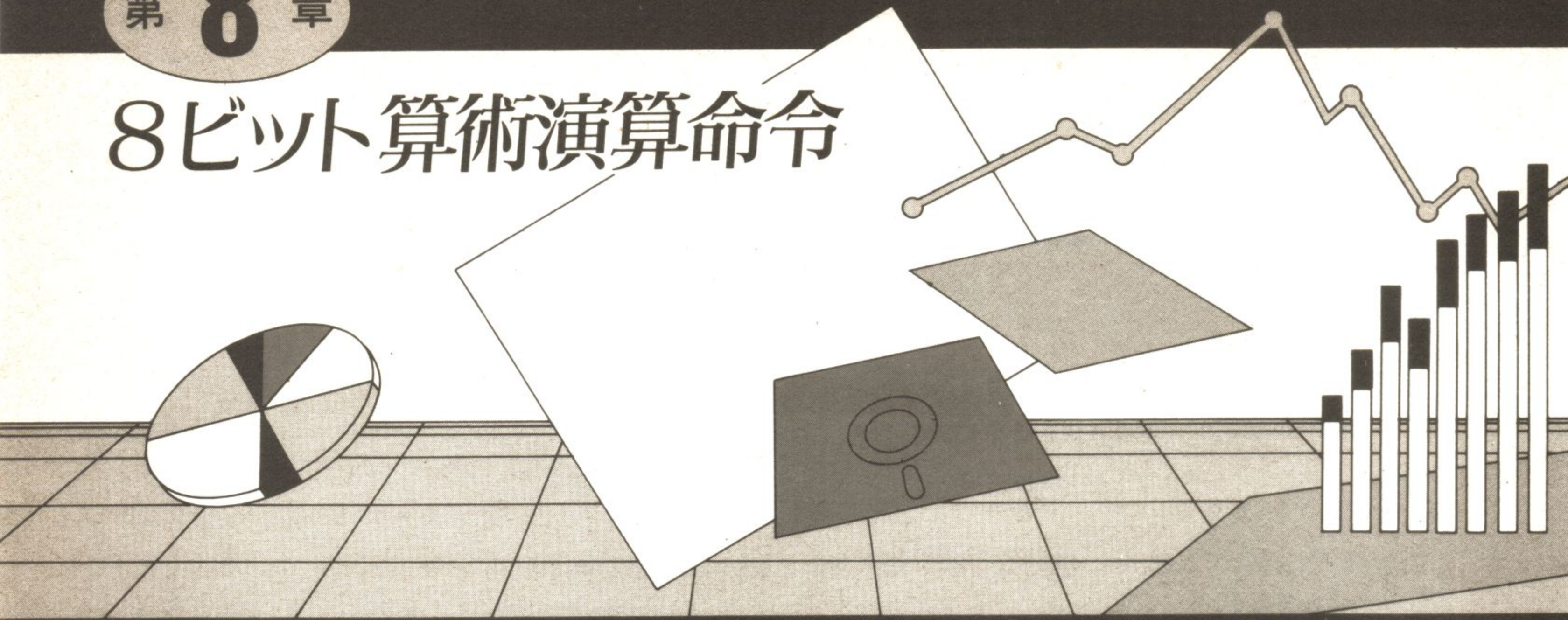
令を知っている事がマシン語への理解をかえって
おくれさせている面もあるかもしれません。

N-BASICの命令と対比することのできる
マシン語の命令も多いのですがこのスタックの考
え方などは、BASICには全く有りませんから、
N-BASICに慣れてしまった頭を完全に切り
換える必要が生じてきます。

『必要は発明の母』などと言いますが、手軽に使
える高級言語やコンパイル言語、そして各種雑誌
やメーカーなどから提供される数々のプログラム
などを見ているとマシン語など勉強するのがいや
になって来ます。ワンボード・マイコンの頃はマ
シン語が勉強し易かったものです！



8ビット算術演算命令



本章からいよいよ演算命令を説明していきます。その中でも本章で紹介するのは、8ビット数値の間で加減算を行うための8ビット算術演算命令です。8ビットの数値で加減算を行う命令とは言っても結局、乗除算命令の無いZ80においてはその代用も行う命令ですからかなり広範囲な応用が考えられます。

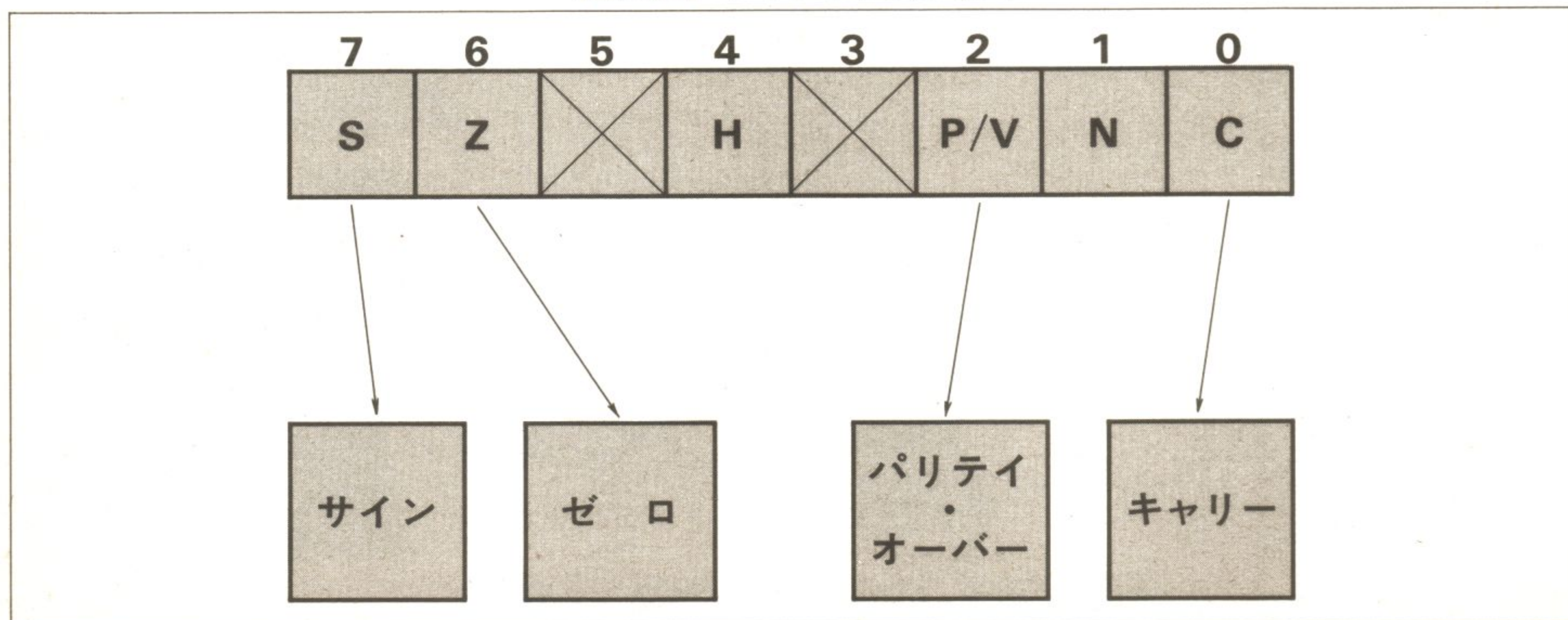
また本章で説明して行く命令からは、全てフラグを変化させてしまいますから、常にフラグ・レジスタの値に注意して行かなければなりません。まず、フラグとはどういうものなのかを考えてみましょう。

フラグについての基礎知識

Z80が、算術・論理演算を行った場合には、その演算結果によって、フラグ・レジスタ(F)が変化します。ここでは、演算のための基礎知識として、フラグの働きについて少し説明したいと思いますが、これは、後に条件判断を利用した条件ジャンプなどを行う場合にも不可欠なものですから、特に注意して読んで欲しいと思います。

フラグ・レジスタ(F)は実際は8ビットのレジスタですが、他のレジスタのように8ビット全てが、まとまった働きをするのではなく、1ビットごとに独立して変化します。ですから8ビットの

《第36図》フラグレジスタ(F)



レジスタと考えるよりも、1ビットのレジスタがいくつかあると考えた方が考え易いかもしれません(第36図)。

フラグ・レジスタの8ビットの内、第5ビットと第3ビットは未定義で使用されませんから実際に使用しているのは6ビットです。また、第4ビットのハーフ・キャリー・フラグ(H)と第1ビットの加算／減算フラグ(N)は、Z80が勝手に使用するもので、この値を調べる事はできませんから今のところ知らなくてもかまわないでしょう。

そうすると残るのは、第7ビットのサイン・フラグ(S)、第6ビットのゼロ・フラグ(Z)、第2ビットのパリティ／オーバー・フラグ(P/V)、第0ビットのキャリー・フラグ(CまたはCY)の4ビットのみとなります。キャリー・フラグはCと書くとCレジスタと間違えおそれがありますので普通はCYと書きます。

この中でも特筆に値するのは、ゼロ・フラグ(Z)と、キャリー・フラグ(CY)の二つで、この二つのフラグさえ理解していればほとんどのプログラムを組めてしまいます。少なくとも一般のゲーム・プログラムなどを組む場合には、この二つのフラグだけを知っておけば十分でしょう。

ですから、このすぐ後に六つのフラグがどのような場合に変化するのかを大まかに説明しますが、特に**ゼロ・フラグ(Z)**と**キャリー・フラグ(CY)**の部分は完全に覚えておいてください。演算命令や条件ジャンプなどを勉強して行く上で必ず助けになることと思います。

ゼロ・フラグ(Z)

ゼロ・フラグ(Z)は、あるレジスタに対して何らかの演算を行った場合にその結果が0になった時1にセットされます。例えば、

$$45 - 45 = 0$$

のような演算を行った場合や

$$45H \wedge 00H = 00H$$

(AND)

のような論理演算によって結果が0になった時にゼロ・フラグは1にセットされます。

キャリー・フラグ(CY)

キャリー・フラグ(CY)の主な働きは、演算命令などを行った場合の桁上り、桁下りを記録しておくためのものです。

Z80では、1バイトの加算・減算を行う事ができますが、例えば減算で

$$30 - 50 = -20$$

を実行して桁下りが生じた場合や、加算を行った結果が1バイトでは足りないほど大きな数になってしまった時などにキャリー・フラグが1にセットされます。

サイン・フラグ(S)

サイン・フラグ(S)は、符号付きの数値を扱った演算の結果が負(マイナス)になった場合に1にセットされます。

1バイトでは普通0~255の数値を扱う事ができますが、**第37図**のように第7ビットを正負の符号専用に使って、そのビットが1のとき-128~-1の数値、そのビットが0の時には、0~127の数値を表し結局-128~+127の数値を扱うこともできます。この最上位ビットと同じものがサイン・フラグ(S)に入ります。

パリティ/オーバーフロー・フラグ(P/V)

このフラグは1ビットで2通りの働きをします。まず第1に論理演算を行った場合のAレジスタのパリティを示します。Z-80の論理演算の結果は全てAレジスタに納められますが、その結果8ビットの中に1であるビットが偶数の時にこのフラグが1となり、1であるビットが奇数の時には、0になります。

第2にオーバーフロー・フラグとしての役割りですが、符号付きの算術演算の結果が、オーバーフローを起こした場合に1にセットされます。

例えば1バイトの符号付き算術演算では、-128

～127の間の数値しか正しく扱うことができませんので、

$$100 + 100 = 200$$

という加算を行った場合には、結果が-128～127の範囲からあふれてしまいますから、このフラグが1にセットされるのです。

普通のプログラムでは、あまり使われることのないフラグでしょう。

ハーフキャリー・フラグ(H)

演算命令を実行した場合に下位4ビットからの桁上りや桁下りなどが生じた場合に1にセットされ、Z80で10進補正のためのDAA命令を行う時にこのフラグが利用されます。

減算フラグ(N)

最後に実行された命令が減算なら1にセットされ加算の場合には0にリセットされます。このフラグもDAA命令による10進補正が正しく行われるように用意されていますので詳しくは後述します。

8ビット算術演算命令

それでは8ビットの算術演算命令を紹介します。便義上、12種に分類させていただきましたが実際は、

- ① **ADD命令** (加算命令)
- ② **ADC命令** (キャリーを含む加算命令)
- ③ **SUB命令** (減算命令)
- ④ **SBC命令** (キャリーを含む減算命令)

の4種のニーモックで表わします。そしてその4種それぞれを、

- i) レジスタを対象とするもの
- ii) 直接の定数を対象とするもの
- iii) メモリ上の数値を対象とするもの

の3種に分けて説明します。

本章で説明した命令についてのニーモニックとマシン語の対応表は、第10章で8ビット算術論理演算命令としてまとめて掲載します。

◎Aレジスタとレジスタの間で

加算を行う命令

Z80では、Aレジスタと他の8ビットのレジスタの内容を加えることができます。加算の結果(答え)はAレジスタに入りますから、命令を実行した後Aレジスタの内容は変化してしまいます。

AレジスタとBレジスタの内容の和をAレジスタに入れたければニーモニックを

ADD A, B

と書きますが、“ADD”と言うのは英語で“加える”という意味です。

N-BASICの

A = A + B

という感覚の命令ですから、すぐわかるでしょう。

この命令では、Aレジスタに、A, B, C, D, E, H, Lの各レジスタの内容を加える事ができ全て1バイトのマシン語にアSEMBルします。

ADD A, A

は無意味な様ですが大へん重要な命令でこれを行う事によってAレジスタの内容を2倍、4倍、8倍にする事ができます。

◎Aレジスタに直接数値を加える命令

BASICでは、

A = A + &H30

のように、Aレジスタに数値を加える命令でニーモニックでは、

ADD A, 30H

のようになります。この場合は、Aレジスタの値に30Hが加えられます。

マシン語では、2バイトの命令になり1バイト目は“C6”，2バイト目には加えたい16進数をそのまま割り当てます。

上記の、

ADD A, 30H

の場合には、マシン語にすると、

C6 30

のように、アSEMBルされます。

実際にマシン語でプログラムを組んで行くと定数を直接Aレジスタに加えたり、比較したりする事はかなり多く行いますから直接数値を与える事ができるのは大へん便利です。

◎Aレジスタに指定した

メモリの内容を加える命令

この命令を使えば、Aレジスタにメモリの内容を加える事ができますが、その場合のメモリのアドレスを指定するためには、HLレジスタ対、IXレジスタ、IYレジスタのいずれかをを用います。ここでは、8080CPUにも使われて、もっとも簡単なHLレジスタ対を使ったアドレス指定を考えてみましょう。

ニーモニックで

```
ADD A, (HL)
```

をすれば、Aレジスタの内容にHLレジスタ対が指すアドレスの内容が加えられます。マシン語では“86”の1バイトです。

ここで一つの例をあげてみましょう。

```
LD HL, E000H
LD A, 03H
LD (HL), A
ADD A, (HL)
```

ずいぶんむだの多いプログラムですが、これを実行する事によってAレジスタに、06Hが代入されることがわかりいただけるでしょうか？

◎Aレジスタにレジスタと

キャリー・フラグを加える命令

この命令は、Aレジスタに他の8ビットのレジスタの内容を加えさらにキャリー・フラグを加えるためのものです。この命令は特に2バイト以上に渡る数値を加算する場合に有効となります。

例えば、DEレジスタ対とBCレジスタ対の和をDEレジスタ対に入れる様な場合には、まず下位のEレジスタとCレジスタの和をEレジスタに入れてから、上位のDレジスタとBレジスタを加えて、Dレジスタに入れればよいのですが、下位の加算で桁上りが生じた場合には、その分も上位に加えてやらなければなりません。

下位の加算は、もちろんADD命令を使うのですが、そこで桁上りが生じた場合には、キャリー・フラグが1にセットされますから、上位の加算の時にキャリー・フラグも加えることによって桁上りの処理ができます。

それでは今の、DEレジスタ対とBCレジスタ対の加算をプログラムにしてみましょう。

```
LD A, E
ADD A, C } 下位バイトを加算
LD E, A
LD A, D } 上位バイトを加算し、
ADC A, B } 下位からの桁上りも加
LD D, A } える
```

このADC命令がキャリー・フラグを含めた加算命令です。ADCは、“ADd with Carry”（キャリーと共に加える）の略です。

◎Aレジスタに直接数値を加え

キャリー・フラグも加える命令

この命令も2バイト以上の加算を行うために有効な命令です。

プログラムの例として、DEレジスタ対の内容に、16進数値の5678Hを加える場合を考えてみましょう。

```
LD A, E } Eレジスタに78Hを
ADD A, 78H } 加えて下位バイトの
LD E, A } 加算を実行
LD A, D } Dレジスタに56Hを
ADC A, 56H } 加え、キャリー・フ
LD D, A } ラグも加える
```

このようにキャリー・フラグは加算命令を実行する場合には、桁上りを表します。つまり、1バイトの数値同士を加えた和が1バイトに入りきれなくなった場合、FFHをこえた場合に1にセットされ、それ以外は0になります。これを利用する事によって、多バイトに渡る数値の加算を行う事が可能となります。

◎Aレジスタに指定したメモリの内容を加え

キャリー・フラグも加える命令

この命令は、Aレジスタにメモリに入っている内容を加え、その上にキャリー・フラグを加える

ものですが、メモリのアドレスを指定するためには、HLレジスタ対、IXレジスタ、またはIYレジスタを使用します。

連続したメモリに入っている2バイト以上に渡る数値を扱う場合には、特に有効な命令となります。

◎Aレジスタからレジスタの内容を引く命令

この命令はAレジスタから8ビットのレジスタの内容を引くための命令で当然の事ですが8ビットの減算を行うために使用します。

例えば、Aレジスタの内容からEレジスタの内容を引いた差をAレジスタに入れるために

```
SUB E
```

と書きます。一見して、Aレジスタを第1オペランドに入れて、

```
SUB A, E
```

と書くのが正しいように思えますが、SUB命令の対象は必ずAレジスタになりますので省略するのが正しいニーモニックです。

SUBとは、英語の“SUBtract”の略で意味は、引くとか減じるとかいう事です。

この命令を使えば、AレジスタからA、B、C、D、E、H、Lの各レジスタの内容を引く事ができ、全て1バイトのマシン語にアSEMBルする事ができます。

```
SUB A
```

はAレジスタを0にするためなどに使います。

◎Aレジスタから直接数値を引く命令

Aレジスタから直接数値を引く場合、例えばAレジスタから12Hを引く時には、

```
SUB 12H
```

を行います。Aレジスタから12Hを引いた差はAレジスタに納められます。

マシン語にアSEMBルした2バイト目は引きたい数値を割り当て、上のニーモニックをマシン語に直すと

```
D 6 · 12
```

になります。

◎Aレジスタから指定した

メモリの内容を引く命令

この命令の場合もメモリの指定に、HLレジスタ対、IXレジスタ又はIYレジスタを用います。

Aレジスタから、E000H番地の内容を引いた差をAレジスタに納めるプログラムを以下に示しますので考えてみてください。

```
LD HL, E000H
```

```
SUB (HL)
```

この例ではHLレジスタ対をアドレスを指定するために用いています。

◎Aレジスタからレジスタの内容と

キャリー・フラグを引く命令

2バイト以上に渡る数値の減算を行う場合に有効な命令で、Aレジスタの内容からCレジスタの内容を引いた上にキャリー・フラグを引くためには、

```
SBC A, C
```

を行います。これは、“SUBtract with Carry”の略でキャリーと共に引くということです。又、SBC命令は16ビットの演算命令にも有りますので第1オペランドの“A”は必ず付けなければなりません。

命令の対象となるレジスタは、A、B、C、D、E、H、Lの各レジスタです。以下にBCレジスタ対からDEレジスタ対を引くプログラムを示します。

```
LD A, C }
SUB E   } 下位バイトを減算
LD C, A }
LD A, B }
SBC A, D } 上位バイトを減算
LD B, A }
```

◎Aレジスタから直接数値を引き

キャリー・フラグを引く命令

この命令は、2バイト以上に渡る減算に使いますがあらかじめ引く数値がわかっている場合に使います。

例えばBCレジスタ対の内容から、1234Hを引く場合には次のようなプログラムとなります。

| | | |
|-----|--------|---------------------------------------|
| LD | A, C | } Cレジスタから34H を減じて下位バイト |
| SUB | 34H | |
| LD | C, A | } の減算を実行 |
| LD | A, B | } Bレジスタから12H を減じて、キャリー・ フラグも減じる |
| SBC | A, 12H | |
| LD | B, A | |

命令は全て2バイトにアセンブルし2バイト目は引きたい16進数を直接割り当てます。上記の

SBC A, 12H

はマシン語では、

DE・12

となります。

◎Aレジスタから指定したメモリの内容を引き キャリー・フラグを引く命令

アドレスの指定には、おなじみのHLレジスタ対、IXレジスタ、IYレジスタを使います。

この命令は、2バイト以上に渡る数値の減算で特にメモリ上の数値を扱う際に使用します。

ここまで読んで来られた皆さんなら、そろそろ御自分でプログラム例を組めるのではないでしょう

1 BYTE

ところで皆さんは、1バイトの数値でどれだけの表現方法があると思いますか？

Z 80のマシン語を扱って行く場合には少なくとも、

- ① 16進数
- ② 2進数
- ③ 10進数 i) 符号無し
 ii) 符号付き

の4種は知っていなければなりません。

そこで、これら4種の表現方法をすぐに比較できるように表にしたものが第37図です。この図では、00H～FFHまでの16進数を、2進数、符号無し10進数、符号付き10進数に変換して比較できる様になっています。

また、この図をPC-8822、PC-8023C等に打ち出すためのプログラムを参考に掲載します。皆さんも、この表をながめることによって、コンピュータ内部の表現形式に親しんで欲しいと思います。

《第37-A図》16進数・2進数・10進数（符号あり、なし）変換表
出力リスト

```

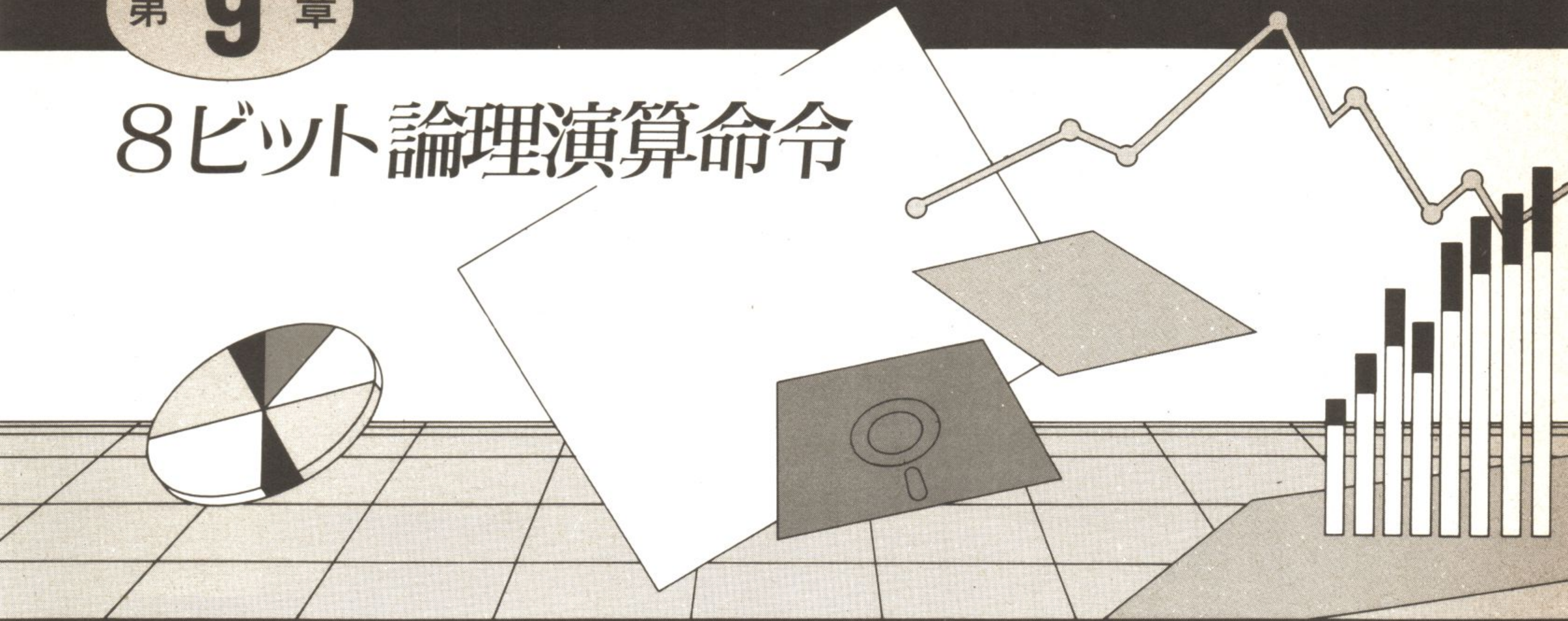
100 DEFINT A-Z
110 LPRINT CHR$( &H1B ); "Q"; CHR$( &H1B ); "I"; CHR$( 12 );
120 FOR K=0 TO 63
130   FOR I=K TO 255 STEP 64
140     LPRINT RIGHT$( "0"+HEX$( I ), 2 ); " ";
150     FOR J=7 TO 0 STEP -1
160       IF 0<( I AND ( 2^J ) ) THEN LPRINT "1"; ELSE LPRINT "0";
170     NEXT J
180     LPRINT " ";:LPRINT USING "####";I;:LPRINT " ";
190     IF 127<I THEN LPRINT USING "####";I-256; ELSE LPRINT USING "####";I;
200     LPRINT " ";
210   NEXT I
220   LPRINT
230 NEXT K

```


《第37-B図》16進数・2進数・10進数(符号あり, なし)変換表

| | | | | | | | | | | | | | | | |
|----|----------|----|----|----|----------|-----|-----|----|----------|-----|------|----|----------|-----|-----|
| 00 | 00000000 | 0 | 0 | 40 | 01000000 | 64 | 64 | 80 | 10000000 | 128 | -128 | C0 | 11000000 | 192 | -64 |
| 01 | 00000001 | 1 | 1 | 41 | 01000001 | 65 | 65 | 81 | 10000001 | 129 | -127 | C1 | 11000001 | 193 | -63 |
| 02 | 00000010 | 2 | 2 | 42 | 01000010 | 66 | 66 | 82 | 10000010 | 130 | -126 | C2 | 11000010 | 194 | -62 |
| 03 | 00000011 | 3 | 3 | 43 | 01000011 | 67 | 67 | 83 | 10000011 | 131 | -125 | C3 | 11000011 | 195 | -61 |
| 04 | 00000100 | 4 | 4 | 44 | 01000100 | 68 | 68 | 84 | 10000100 | 132 | -124 | C4 | 11000100 | 196 | -60 |
| 05 | 00000101 | 5 | 5 | 45 | 01000101 | 69 | 69 | 85 | 10000101 | 133 | -123 | C5 | 11000101 | 197 | -59 |
| 06 | 00000110 | 6 | 6 | 46 | 01000110 | 70 | 70 | 86 | 10000110 | 134 | -122 | C6 | 11000110 | 198 | -58 |
| 07 | 00000111 | 7 | 7 | 47 | 01000111 | 71 | 71 | 87 | 10000111 | 135 | -121 | C7 | 11000111 | 199 | -57 |
| 08 | 00001000 | 8 | 8 | 48 | 01001000 | 72 | 72 | 88 | 10001000 | 136 | -120 | C8 | 11001000 | 200 | -56 |
| 09 | 00001001 | 9 | 9 | 49 | 01001001 | 73 | 73 | 89 | 10001001 | 137 | -119 | C9 | 11001001 | 201 | -55 |
| 0A | 00001010 | 10 | 10 | 4A | 01001010 | 74 | 74 | 8A | 10001010 | 138 | -118 | CA | 11001010 | 202 | -54 |
| 0B | 00001011 | 11 | 11 | 4B | 01001011 | 75 | 75 | 8B | 10001011 | 139 | -117 | CB | 11001011 | 203 | -53 |
| 0C | 00001100 | 12 | 12 | 4C | 01001100 | 76 | 76 | 8C | 10001100 | 140 | -116 | CC | 11001100 | 204 | -52 |
| 0D | 00001101 | 13 | 13 | 4D | 01001101 | 77 | 77 | 8D | 10001101 | 141 | -115 | CD | 11001101 | 205 | -51 |
| 0E | 00001110 | 14 | 14 | 4E | 01001110 | 78 | 78 | 8E | 10001110 | 142 | -114 | CE | 11001110 | 206 | -50 |
| 0F | 00001111 | 15 | 15 | 4F | 01001111 | 79 | 79 | 8F | 10001111 | 143 | -113 | CF | 11001111 | 207 | -49 |
| 10 | 00010000 | 16 | 16 | 50 | 01010000 | 80 | 80 | 90 | 10010000 | 144 | -112 | D0 | 11010000 | 208 | -48 |
| 11 | 00010001 | 17 | 17 | 51 | 01010001 | 81 | 81 | 91 | 10010001 | 145 | -111 | D1 | 11010001 | 209 | -47 |
| 12 | 00010010 | 18 | 18 | 52 | 01010010 | 82 | 82 | 92 | 10010010 | 146 | -110 | D2 | 11010010 | 210 | -46 |
| 13 | 00010011 | 19 | 19 | 53 | 01010011 | 83 | 83 | 93 | 10010011 | 147 | -109 | D3 | 11010011 | 211 | -45 |
| 14 | 00010100 | 20 | 20 | 54 | 01010100 | 84 | 84 | 94 | 10010100 | 148 | -108 | D4 | 11010100 | 212 | -44 |
| 15 | 00010101 | 21 | 21 | 55 | 01010101 | 85 | 85 | 95 | 10010101 | 149 | -107 | D5 | 11010101 | 213 | -43 |
| 16 | 00010110 | 22 | 22 | 56 | 01010110 | 86 | 86 | 96 | 10010110 | 150 | -106 | D6 | 11010110 | 214 | -42 |
| 17 | 00010111 | 23 | 23 | 57 | 01010111 | 87 | 87 | 97 | 10010111 | 151 | -105 | D7 | 11010111 | 215 | -41 |
| 18 | 00011000 | 24 | 24 | 58 | 01011000 | 88 | 88 | 98 | 10011000 | 152 | -104 | D8 | 11011000 | 216 | -40 |
| 19 | 00011001 | 25 | 25 | 59 | 01011001 | 89 | 89 | 99 | 10011001 | 153 | -103 | D9 | 11011001 | 217 | -39 |
| 1A | 00011010 | 26 | 26 | 5A | 01011010 | 90 | 90 | 9A | 10011010 | 154 | -102 | DA | 11011010 | 218 | -38 |
| 1B | 00011011 | 27 | 27 | 5B | 01011011 | 91 | 91 | 9B | 10011011 | 155 | -101 | DB | 11011011 | 219 | -37 |
| 1C | 00011100 | 28 | 28 | 5C | 01011100 | 92 | 92 | 9C | 10011100 | 156 | -100 | DC | 11011100 | 220 | -36 |
| 1D | 00011101 | 29 | 29 | 5D | 01011101 | 93 | 93 | 9D | 10011101 | 157 | -99 | DD | 11011101 | 221 | -35 |
| 1E | 00011110 | 30 | 30 | 5E | 01011110 | 94 | 94 | 9E | 10011110 | 158 | -98 | DE | 11011110 | 222 | -34 |
| 1F | 00011111 | 31 | 31 | 5F | 01011111 | 95 | 95 | 9F | 10011111 | 159 | -97 | DF | 11011111 | 223 | -33 |
| 20 | 00100000 | 32 | 32 | 60 | 01100000 | 96 | 96 | A0 | 10100000 | 160 | -96 | E0 | 11100000 | 224 | -32 |
| 21 | 00100001 | 33 | 33 | 61 | 01100001 | 97 | 97 | A1 | 10100001 | 161 | -95 | E1 | 11100001 | 225 | -31 |
| 22 | 00100010 | 34 | 34 | 62 | 01100010 | 98 | 98 | A2 | 10100010 | 162 | -94 | E2 | 11100010 | 226 | -30 |
| 23 | 00100011 | 35 | 35 | 63 | 01100011 | 99 | 99 | A3 | 10100011 | 163 | -93 | E3 | 11100011 | 227 | -29 |
| 24 | 00100100 | 36 | 36 | 64 | 01100100 | 100 | 100 | A4 | 10100100 | 164 | -92 | E4 | 11100100 | 228 | -28 |
| 25 | 00100101 | 37 | 37 | 65 | 01100101 | 101 | 101 | A5 | 10100101 | 165 | -91 | E5 | 11100101 | 229 | -27 |
| 26 | 00100110 | 38 | 38 | 66 | 01100110 | 102 | 102 | A6 | 10100110 | 166 | -90 | E6 | 11100110 | 230 | -26 |
| 27 | 00100111 | 39 | 39 | 67 | 01100111 | 103 | 103 | A7 | 10100111 | 167 | -89 | E7 | 11100111 | 231 | -25 |
| 28 | 00101000 | 40 | 40 | 68 | 01101000 | 104 | 104 | A8 | 10101000 | 168 | -88 | E8 | 11101000 | 232 | -24 |
| 29 | 00101001 | 41 | 41 | 69 | 01101001 | 105 | 105 | A9 | 10101001 | 169 | -87 | E9 | 11101001 | 233 | -23 |
| 2A | 00101010 | 42 | 42 | 6A | 01101010 | 106 | 106 | AA | 10101010 | 170 | -86 | EA | 11101010 | 234 | -22 |
| 2B | 00101011 | 43 | 43 | 6B | 01101011 | 107 | 107 | AB | 10101011 | 171 | -85 | EB | 11101011 | 235 | -21 |
| 2C | 00101100 | 44 | 44 | 6C | 01101100 | 108 | 108 | AC | 10101100 | 172 | -84 | EC | 11101100 | 236 | -20 |
| 2D | 00101101 | 45 | 45 | 6D | 01101101 | 109 | 109 | AD | 10101101 | 173 | -83 | ED | 11101101 | 237 | -19 |
| 2E | 00101110 | 46 | 46 | 6E | 01101110 | 110 | 110 | AE | 10101110 | 174 | -82 | EE | 11101110 | 238 | -18 |
| 2F | 00101111 | 47 | 47 | 6F | 01101111 | 111 | 111 | AF | 10101111 | 175 | -81 | EF | 11101111 | 239 | -17 |
| 30 | 00110000 | 48 | 48 | 70 | 01110000 | 112 | 112 | B0 | 10110000 | 176 | -80 | F0 | 11110000 | 240 | -16 |
| 31 | 00110001 | 49 | 49 | 71 | 01110001 | 113 | 113 | B1 | 10110001 | 177 | -79 | F1 | 11110001 | 241 | -15 |
| 32 | 00110010 | 50 | 50 | 72 | 01110010 | 114 | 114 | B2 | 10110010 | 178 | -78 | F2 | 11110010 | 242 | -14 |
| 33 | 00110011 | 51 | 51 | 73 | 01110011 | 115 | 115 | B3 | 10110011 | 179 | -77 | F3 | 11110011 | 243 | -13 |
| 34 | 00110100 | 52 | 52 | 74 | 01110100 | 116 | 116 | B4 | 10110100 | 180 | -76 | F4 | 11110100 | 244 | -12 |
| 35 | 00110101 | 53 | 53 | 75 | 01110101 | 117 | 117 | B5 | 10110101 | 181 | -75 | F5 | 11110101 | 245 | -11 |
| 36 | 00110110 | 54 | 54 | 76 | 01110110 | 118 | 118 | B6 | 10110110 | 182 | -74 | F6 | 11110110 | 246 | -10 |
| 37 | 00110111 | 55 | 55 | 77 | 01110111 | 119 | 119 | B7 | 10110111 | 183 | -73 | F7 | 11110111 | 247 | -9 |
| 38 | 00111000 | 56 | 56 | 78 | 01111000 | 120 | 120 | B8 | 10111000 | 184 | -72 | F8 | 11111000 | 248 | -8 |
| 39 | 00111001 | 57 | 57 | 79 | 01111001 | 121 | 121 | B9 | 10111001 | 185 | -71 | F9 | 11111001 | 249 | -7 |
| 3A | 00111010 | 58 | 58 | 7A | 01111010 | 122 | 122 | BA | 10111010 | 186 | -70 | FA | 11111010 | 250 | -6 |
| 3B | 00111011 | 59 | 59 | 7B | 01111011 | 123 | 123 | BB | 10111011 | 187 | -69 | FB | 11111011 | 251 | -5 |
| 3C | 00111100 | 60 | 60 | 7C | 01111100 | 124 | 124 | BC | 10111100 | 188 | -68 | FC | 11111100 | 252 | -4 |
| 3D | 00111101 | 61 | 61 | 7D | 01111101 | 125 | 125 | BD | 10111101 | 189 | -67 | FD | 11111101 | 253 | -3 |
| 3E | 00111110 | 62 | 62 | 7E | 01111110 | 126 | 126 | BE | 10111110 | 190 | -66 | FE | 11111110 | 254 | -2 |
| 3F | 00111111 | 63 | 63 | 7F | 01111111 | 127 | 127 | BF | 10111111 | 191 | -65 | FF | 11111111 | 255 | -1 |

8ビット論理演算命令



8ビット論理演算命令

第8章では、8ビットの数値間で加算・減算を行うための8ビット算術演算命令を紹介しましたが、本章では、同じく8ビットの数値間で論理演算を行うための8ビット論理演算命令を説明します。

Z80では、論理和(OR)、論理積(AND)、及び、排他的論理和(XOR)を、1ステップの命令によって求める事ができます。

N-BASICなどを使っている場合には、論理演算など不要のごとく思える場合もあるかもしれませんが、数少ない命令しか用意されていないマシン語で採用されている事でもわかるように必要不可欠な命令ですので、ぜひとも使い方を覚えて欲しいと思います。

最後には、論理命令の使用例として、あの有名なギャラクシアン……を、一匹だけ画面上に表示・消去するための例題を紹介しますので、ぜひ何かに応用してみてください。また、このプログラムでは、初めてインデックス・レジスタを用いたインデックス・アドレス指定を行いましたので、同時にそちらの方もマスターしてください。

◎Aレジスタとレジスタの間で

論理積ANDを求める命令

Aレジスタの内容8ビットと8ビットのレジスタの間で1ビットごとに、ANDを求めるための

命令です。

例えば、Aレジスタに55Hが入っており、Bレジスタに0FHが入っている場合にAND命令を実行する場合には、

AND B

を行います。結果は必ずAレジスタに入りますのでAは省略しなければなりません。

上の論理演算を実行すると、

```

0101 0101.....55H
AND) 0000 1111.....0FH

```

```

0000 0101.....05H

```

となり、Aレジスタには、結果の05Hが入ります。

この様に、ANDとは、両者が1のビットのみ1となり、他のビットは全て0になる演算の事なのです。

上記のように0FHとANDをとる事は、上位4ビットを無条件に0にマスクしてしまう事になります。

◎Aレジスタと指定するメモリの内容との間で 論理積ANDを求める命令

メモリの指定には、よく登場するHLレジスタ対、IYレジスタ、IXレジスタのいずれかを使用する事ができますが、ここでは最もポピュラーなHLレジスタ対の場合を例にとってみましょう。

例えばAレジスタとE000H番地の内容との間で求めた論理積を、Aレジスタに入れるのには、


```
LD    HL, E 0 0 0 H
AND   (HL)
```

を実行します。

同じ命令を、IXレジスタを用いて考えると、

```
LD    IX, E 0 0 0 H
AND   (IX + 0 0 H)
```

となります。2ステップ目の

```
IX + 0 0 H
```

は、指定するアドレスが、IXレジスタの内容に00Hを加えたアドレスという事で、もし、

```
AND   (IX + 0 5 H)
```

の場合には、E005H番地を指定した事になります。

◎Aレジスタと直接に与える数値の間で

論理積ANDを求める命令

Aレジスタと直接に指定する8ビットの数値の間で、1ビットごとにANDを求めるための命令です。

例えば、Aレジスタの第2ビットから第5ビットまでの真中4ビットを0にマスクするためには、

```
AND   C 3 H
```

を実行します。

16進数のC3Hは2進数では、

```
1 1 0 0   0 0 1 1
```

ですから、どんな数値がAレジスタに入っているも、真中の4ビットは必ず0になります。

AレジスタがFFHの場合を考えてみましょう。

```
1 1 1 1   1 1 1 1 ..... FFH
AND) 1 1 0 0   0 0 1 1 ..... C 3 H
```

```
1 1 0 0   0 0 1 1 ..... C 3 H
```

この場合には、両者共に1のビットは、両端の2ビットずつしかありませんから、結局真中の4ビットは0にマスクされてしまいます。

この命令を、マシン語にアセンブルした場合は2バイトになり、1バイト目はE6Hとなり、2バイト目にはANDを求めるために直接与える数値を直接割り当てます。

上記の

```
AND   C 3 H
```

をアセンブルする場合には、

```
E 6 · C 3
```

の2バイトになります。

◎Aレジスタとレジスタの間で

論理和ORを求める命令

Aレジスタの内容8ビットと、8ビットのレジスタの間で、1ビットごとにORをとるための命令です。

例えば、Bレジスタに55Hが入っており、Cレジスタに0FHが入っている場合に、B、C両レジスタの論理和をAレジスタに求めるためには、以下のようにします。

```
LD    A, B
OR    C
```

論理和は必ずAレジスタを介して求めなければなりませんから、LD命令でBレジスタの内容をAレジスタに持って来た後、AレジスタとCレジスタのORをとります。

また、このOR命令もAを省略しなければなりません。

ちなみに上記の命令を実行すると、

```
0 1 0 1   0 1 0 1 ..... 5 5 H
OR) 0 0 0 0   1 1 1 1 ..... 0 F H
```

```
0 1 0 1   1 1 1 1           5 F H
```

となり、Aレジスタには5FHが入ります。

この様にORというのは、どちらか一つが1の場合には、答えも1となるものですから、AND命令とは逆に8ビットの中の幾つかのビットを無条件に1にしたい時などに用いるための命令なのです。

◎Aレジスタと指定するメモリの内容との間で

論理和ORを求める命令

Aレジスタとメモリの間で求めた論理和をAレジスタに入れる命令ですが、メモリのアドレス指定には、HLレジスタ対、IXレジスタ、IYレジスタの中のいずれかを用います。

例えばIYレジスタを用いて、E000H番地とE001H番地の内容の論理和を求めて、さらにその結果とE002H番地の内容の論理和をAレジスタに収納するプログラム例を組むと次のようになります。


```
LD    IY, E000H
LD    A, (IY+00H)
OR     (IY+01H)
OR     (IY+02H)
```

IX, IYのインデックスレジスタを用いるとこのような応用が可能になります。上記の4ステップのプログラムをマシン語にアセンブルすると、

```
FD·21·00·E0·FD·7E·00
FD·B6·01·FD·B6·02
```

の13バイトになります。

◎Aレジスタとレジスタの間で

論理和ORを求める命令

Aレジスタと、直接に指定する8ビットの数値の間で、1ビットごとにORを求めるための命令です。

例えば、Aレジスタの第6ビットのみを1にセットしたければ、

```
OR    40H
```

を行います。これによって、Aレジスタにどんな値が入っていても、他のビットに変化を与える事無く、第6ビットのみを1セットする事ができます。

Aレジスタに、AAHが入っている場合を考えると、

```
1010 1010 .....AAH
OR) 0100 0000 .....40H
```

```
1110 1010 .....EAH
```

となり、第6ビットが1にセットされた事がよくわかると思います。

この命令は、マシン語にアセンブルする段階で2バイトになり、1バイト目はF6Hですが、2バイト目には、ORをとるために与える数値を直接割り当てます。

◎Aレジスタとレジスタの間で

排他的論理和XORを求める命令

PC-8001を使って、グラフィック画面を扱う場合などには、よくこのXORが使われます。

N-BASICのPUT@命令などの命令を使って宇宙船の絵を星の上に重ね描きしたり、元の

星を消さない様に宇宙船のみを消したい場合には、オプションとして、PSETやPRESETではなく、XORを利用します。

このXORとは、一方が1でもう一方が0の場合にのみ1となって、両者が1又は両者が0の場合、つまり両方が同じ場合には0になります。

例として、Aレジスタに入っているC3Hと、Bレジスタの55HでXORをとる場合を考えてみると命令は、

```
XOR    B
```

となり、実際には、

```
1100 0011 .....C3H
XOR) 0101 0101 .....55H
```

```
1001 0110 .....96H
```

の演算が行われて、結果の96HがAレジスタに収納されます。

この96Hに再度

```
XOR    B
```

を行って、55HとのXORをとってみると、

```
1001 0110 .....96H
XOR) 0101 0101 .....55H
```

```
1100 0011 .....C3H
```

のように始めのC3Hにもどります。この原理を利用しているため、先程の宇宙船と星の例でも、星の上を重ねて宇宙船が通過しても、前の星が消えずに残っているのです。

また、マシン語プログラム中では、よく、

```
XOR    A
```

という命令が使われますが、これはAレジスタの内容を00Hにクリアする場合によく用いられます。

```
LD     A, 00H
```

としても同じ事なのですが、前者の方が1バイトで済みますので、覚えておくと、なにかと便利ではないでしょうか？

◎Aレジスタと直接に与える数値の間で

排他的論理和XORを求める命令

Aレジスタの内容と、直接に与える8ビットの数値の間で1ビットごとにXORをとるための命令です。

実際にこの命令を使用している例として、ギャラクシアンをテレビ画面中央付近に表示するプログラムを紹介しましょう。

プログラム例については後述しますが、本章ではIXレジスタをアドレス指定に用いるインデックス・アドレス指定を行っています。

◎Aレジスタと直接に与える数値の間で 排他的論理和XORを求める命令

Aレジスタの内容と、直接に与える8ビットの数値の間で1ビットごとにXORをとるための命令です。

例えば、Aレジスタと55HでXORをとるためには、

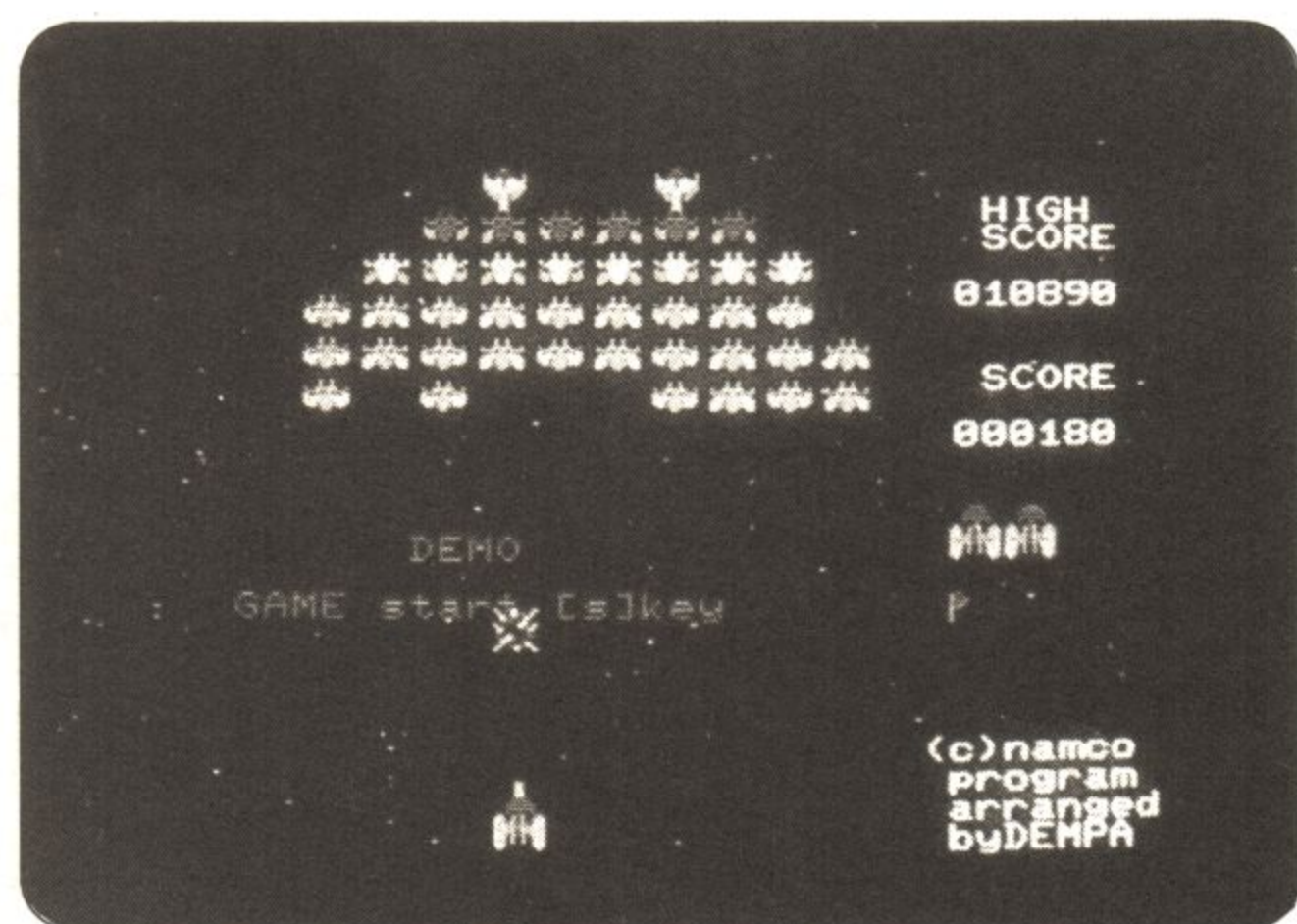
XOR 55H

を行い、マシン語では、EE・55となります。

最後になりましたが、XORとは、eXclusive-ORの略となっています。

ギャラクシアンの点滅プログラム

論理演算命令の使用例として、PC-8001のドット・グラフィックを用いた「ギャラクシアン表示プログラム」を紹介します。



▲DEMPAマイコンソフト (©namco)
PC-8001mkII版・ギャラクシアン

このプログラムは、テレビ画面中央に、ギャラクシアンが1匹(?)出現して点滅するプログラムで、メイン・ルーチンは、N-BASICで組んでありますが、ギャラクシアンの表示と消去のためにE000H番地以降にマシン語のサブルーチンを用意して、XORを用いて表示しています。

このプログラムでは、1本のマシン語サブルーチンのみで、表示・消去の両方を行っています。これもXORを利用しているので可能なのです。

1回目のループで、ギャラクシアンを表示し、2回目には、XORを用いて同じ場所にギャラクシアンを表示する事によって消去しています。

説明だけでは、わかりにくいと思いますので、実例をあげてみましょう。

マシン語サブルーチンについて

何も表示されていない場合、グラフィック・モードのビデオ・ラムには、00Hが入っています。

この上に、ギャラクシアンの左肩である“目”のパターンをXORを用いて表示すると、

```
0000 0000.....00H
XOR) 1000 1100.....8CH
```

1000 1100.....8CH
となり、8CHのパターンである“目”が画面上に出現するのです。

逆に消去する場合には、すでに8CHのパターンが出現している部分に、同じ8CHをXORすれば

```
1000 1100.....8CH
XOR) 1000 1100.....8CH
```

0000 0000.....00H
となり、画面上の8CHが00Hとなって、消去したのと同じ事になります。

この様にE000H番地からのマシン語サブルーチンでは、横10ドット×縦8ドットの10キャラクタのグラフィック・パターンとビデオ・ラムの内容をXORをとってその結果をビデオ・ラム上の同じ位置に送り込む役割りを果たしています。

また、IXレジスタやIYレジスタなどのインデックスレジスタを用いてアドレスを指定する場合には、前章で掲載したような、符号付きの8ビットとしてアSEMBルします。

220行からマシン語のデータが組み込んでありますが、マシン語データの右にREM文として、全てニーモニックを入れておきましたので、ぜひ解析して応用してみてください。

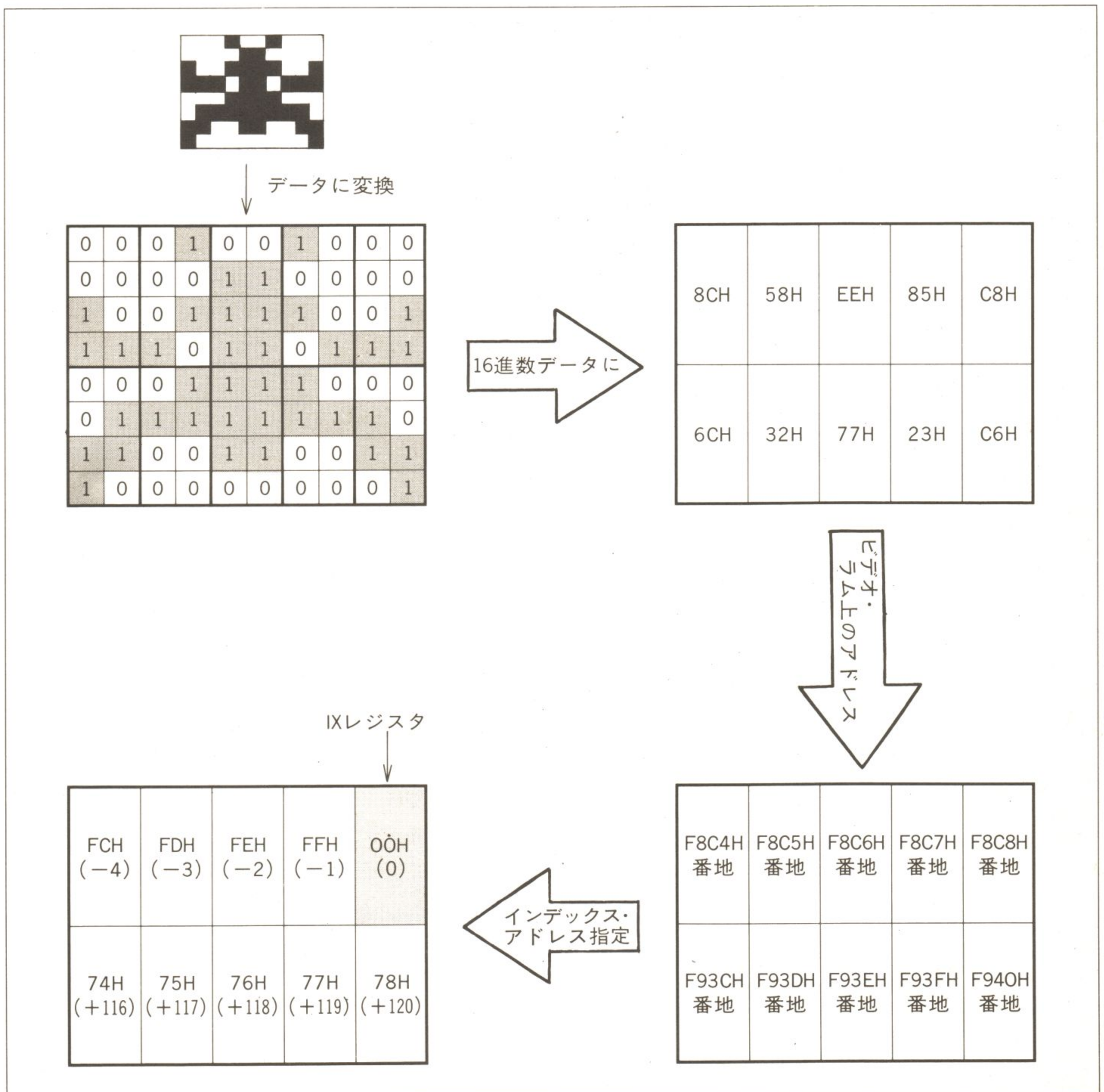
データは3行ごとに区切っていますが、初めの3行とほとんど同じ動作をくり返しているだけ

ですからあとはすぐ解析できると思います。

このプログラムは、打ち込んで実行しても余り楽しいものではなく、マシン語入門のテキスト用に組んだものですから、今までに登場された命令しか使っておりません。ですから非常にむだが多くまた、ギャラクシアンデザインなどもかなりいい加減なものですが、反面わかり易くREM文なども多用しています。

実行させるだけの場合は、200行以外のREM文は全てはずしてもかまいません。

《第38図》ギャラクシアン表示データ



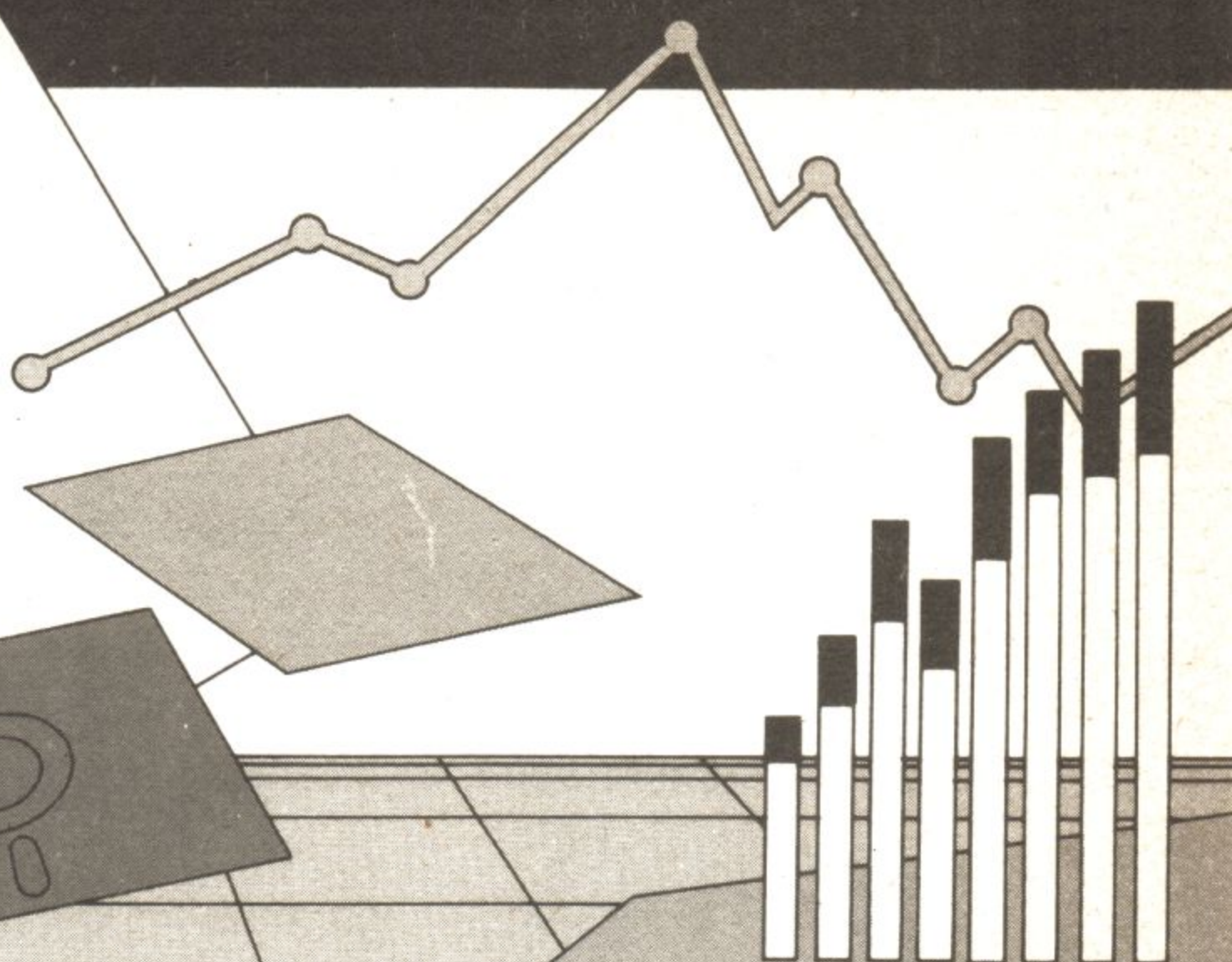
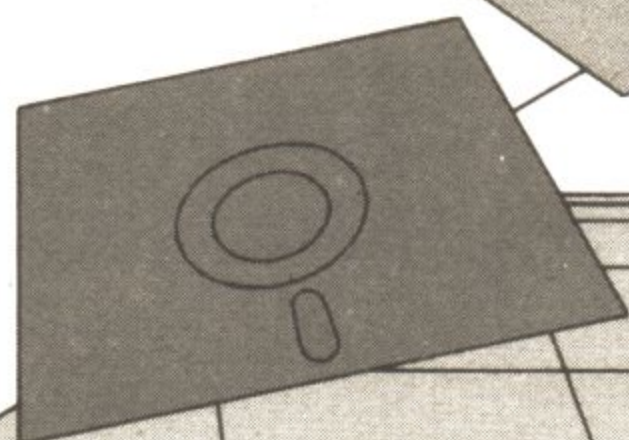
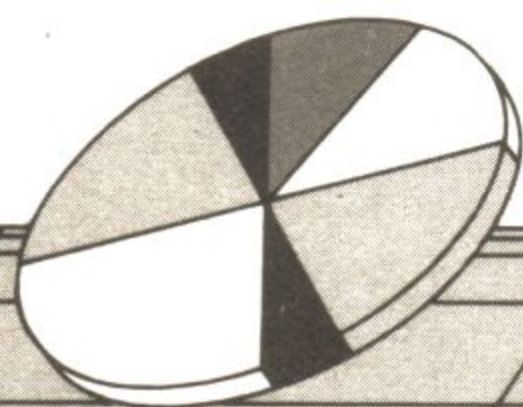
《第39図》ギャラクシアン表示プログラム

```

100 ' ****
110 ' *** XOR GALAXIANS FOR MACHINE CODE 9 ***
120 ' *** BY KIYOSHI KAWAMURA IN FORESIGHT ***
130 ' ****
140
150 CLEAR 300,&HFFFF:DEFINT A-Z:GOSUB200:DEFUSR=&HE000
160 WIDTH 80,25:CONSOLE 0,25,0,0:COLOR 0,0,1:PRINT CHR$(12);
170
180 I=USR(0):FOR W=1 TO 500:NEXT:GOTO 180 'MAIN LOOP
190
200 ' *** READ DATA & POKE MACHINE CODE SUB ***
210
220 FOR A=&HE000 TO &HE054:READ M$:POKE A,VAL("&H"+M$):NEXT:RETURN
230
240 '   マシンゴ データ           :   ニーモニック
250 DATA DD,21,C8,F8           : E000 LD      IX,F8C8H
260                               :
270 DATA 3E,8C                 : E004 LD      A,8CH
280 DATA DD,AE,FC              : E006 XOR     (IX-4)
290 DATA DD,77,FC              : E009 LD      (IX-4),A
300                               :
310 DATA 3E,58                 : E00C LD      A,58H
320 DATA DD,AE,FD              : E00E XOR     (IX-3)
330 DATA DD,77,FD              : E011 LD      (IX-3),A
340                               :
350 DATA 3E,EE                 : E014 LD      A,EEH
360 DATA DD,AE,FE              : E016 XOR     (IX-2)
370 DATA DD,77,FE              : E019 LD      (IX-2),A
380                               :
390 DATA 3E,85                 : E01C LD      A,85H
400 DATA DD,AE,FF              : E01E XOR     (IX-1)
410 DATA DD,77,FF              : E021 LD      (IX-1),A
420                               :
430 DATA 3E,C8                 : E024 LD      A,C8H
440 DATA DD,AE,00              : E026 XOR     (IX+0)
450 DATA DD,77,00              : E029 LD      (IX+0),A
460                               :
470 DATA 3E,6C                 : E02C LD      A,6CH
480 DATA DD,AE,74              : E02E XOR     (IX+116)
490 DATA DD,77,74              : E031 LD      (IX+116),A
500                               :
510 DATA 3E,32                 : E034 LD      A,32H
520 DATA DD,AE,75              : E036 XOR     (IX+117)
530 DATA DD,77,75              : E039 LD      (IX+117),A
540                               :
550 DATA 3E,77                 : E03C LD      A,77H
560 DATA DD,AE,76              : E03E XOR     (IX+118)
570 DATA DD,77,76              : E041 LD      (IX+118),A
580                               :
590 DATA 3E,23                 : E044 LD      A,23H
600 DATA DD,AE,77              : E046 XOR     (IX+119)
610 DATA DD,77,77              : E049 LD      (IX+119),A
620                               :
630 DATA 3E,C6                 : E04C LD      A,C6H
640 DATA DD,AE,78              : E04E XOR     (IX+120)
650 DATA DD,77,78              : E051 LD      (IX+120),A
660                               :
670 DATA C9                   : E054 RET

```


インクリメント命令 /ディクリメント命令



本章では、8ビット算術演算命令の中で残っている、インクリメント命令とディクリメント命令を紹介したいと思います。両者共に数多く使用される命令ですが、非常に簡単な命令です。

また、本章で8ビットの演算命令は全て説明を終了しますので、短いプログラム例で、ちょっとした計算を行ってみたいと思います。

単純な計算で、コンピュータの醍醐味からは程遠いですが、ぜひ実際に走らせてマシン語の楽しさを味わってみてください。

インクリメント命令

/ディクリメント命令

インクリメント命令は、各レジスタ(対)やメモリ上の数値に1を加えるための命令で、主としてくり返し作業(ループ)中で用いられます。

逆に、ディクリメント命令は、各レジスタ(対)やメモリ上の数値から1を減じるための命令で、主にくり返し作業(ループ)で、何回作業を行ったかを調べるカウンタなどとして用いる事が多いようです。

ループのつくり方については、条件ジャンプの時に説明しますが、フローチャートだけを第40図に示しておきます。このフローチャートのように、ループ回数をカウントするループ・カウンタ用のレジスタを制御するのが、ディクリメント命令の最も一般的な使用方法でしょう。

また、インクリメント命令・ディクリメント命令共算術演算命令ですから、フラグに影響を与え、変化したフラグの判断は、後に登場する条件ジャンプによって行います。

なおインクリメント命令、ディクリメント命令のニーモニックは、INC、DECですが、これはINCrement、DECrementの略です。

◎レジスタの内容に1を加える

インクリメント命令

レジスタの内容を1だけ増加する簡単な命令ですが、この命令はかなり頻繁に使用され、無いと大変不便です。

一見して、加算命令で1を加えても同じ様に思えるかも知れませんが、Aレジスタの内容に1を加えるだけならば、直接数値を加える事ができますから、

```
ADD A, 01H
```

でも良いのですが、他のレジスタたとえばBレジスタに1を加えたければ、

```
LD A, B
```

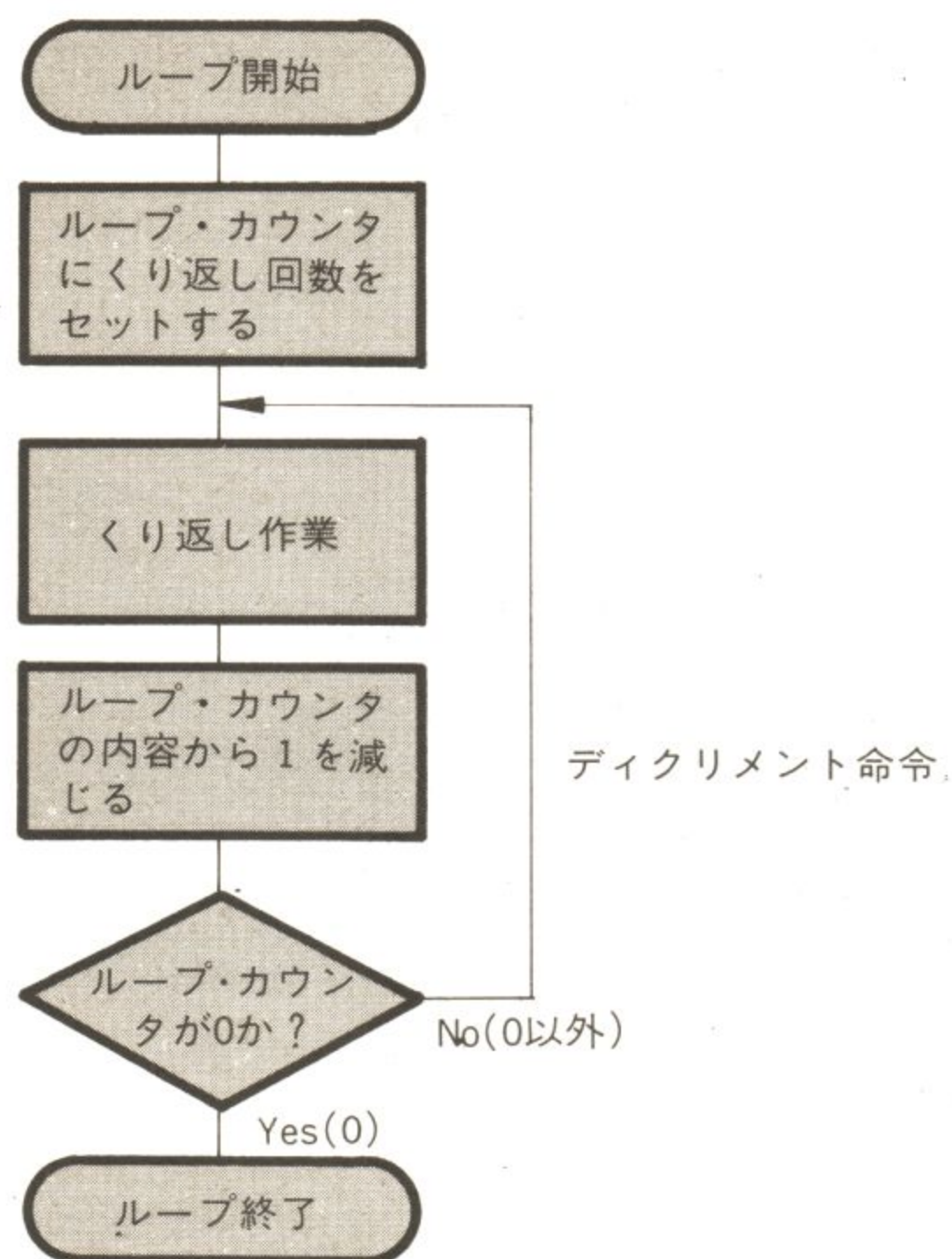
```
ADD A, 01H
```

```
LD B, A
```

の様に一度他のレジスタを使わなければならない、非常に見づらい、プログラムになってしまいます。

レジスタの値に1を加えるためのインクリメント命令は、1バイトのマシン語となり、A、B、C、D、E、H、Lの5個のレジスタに対して有

《第40図》 ループのフローチャート



効です。

例をあげますと、Aレジスタの内容を1増すためには、

INC A

を行い、Hレジスタの内容に3を加えるには、

INC H

INC H

INC H

と、インクリメント命令を続けて三回行なえば良いのです。インクリメント命令の必要性は、N-BASICのプログラム中などに、

A = A + 1

などが多く使われている事でもわかると思います。

◎指定メモリの内容に1を加える

インクリメント命令

メモリを指定して、そのメモリの内容に1を加えるための命令で、メモリの指定は、HLレジスタ対、IXレジスタ、IYレジスタのいずれかを用います。

アドレス指定にHLレジスタ対を用いた場合には1バイトのマシン語にアセンブルしますが、IX又はIYのインデックス・レジスタでアドレス指定を行う場合には、3バイトのマシン語命令になります。

例として、E000H番地からE004H番地までの内容を全て、1ずつ増加するためのプログラムを次に示しますが、アドレス指定にはIXレジスタを用いました。

LD IX, E000H

INC (IX+00H)

INC (IX+01H)

INC (IX+02H)

INC (IX+03H)

INC (IX+04H)

このプログラム例はインデックス・レジスタを用いると簡単ですが、HLレジスタ対では多少面倒になります。

◎レジスタの内容から1を減じる

ディクリメント命令

各レジスタの内容を-1するための命令で、Z80によって、くり返し（ループ）処理を行う場合には、必ずと言って良いほど用いられる命令ですので、通常のプログラム中には、かなり頻繁に登場する重要な命令です。

例えば、

DEC B

を行う事によって、Bレジスタの内容から1を減じる事ができ、同じ様にして、A, B, C, D, E, H, Lの各レジスタの内容から1を減じる事ができます。

又、このディクリメント命令は、全て1バイトのマシン語にアセンブルします。

ディクリメント命令の重要性は、N-BASIC等の、FOR~NEXTループを使う頻度が高い事でもわかると思いますが、もちろん減算命令を用いて1を減じても同じ事です。このように同じ事を行うための命令が用意してあっても、頻繁に使われる命令では、少しでも省メモリで又、見易いプログラムを組むための配慮がされているのです。

◎指定メモリの内容から1を減じる

ディクリメント命令

HLレジスタ対、IXレジスタ、IYレジスタによって指定したメモリの内容から1を減じるための命令で、アドレス指定にHLレジスタ対を使

用した場合は1バイトに、インデックス・レジスタを使用した場合には3バイトになります。

例えば、E 0 0 0 H番地の内容から1を減じる場合には、

```
LD    HL, E 0 0 0 H
DEC   (HL)
```

又は、

```
LD    IX, E 0 0 0 H
DEC   (IX + 0 0 H)
```

のように行います。

プログラム例

以上で、8ビットの算術・論理演算命令を全て説明しましたので、それらの命令を使った応用プログラムを紹介したいと思います。

又、第44図に、8ビット演算命令のニーモニック→マシン語の対応表を掲載しておきますので、御自分のプログラムを組む時の参考にしてください。

第41図のプログラム例を見てください。これは、Aレジスタの内容に1を加えて5倍した後3を減じたものをAレジスタにもどすためのプログラム例です。

$$A \leftarrow (A + 1) \times 5 - 3$$

まず、Aレジスタの内容に1を加えるために、今月説明したインクリメント命令を用いて、

```
INC A
```

を行います。

次に、Aレジスタの内容を5倍するのですが、そのために、わざわざ乗算のためのサブルーチンを用いるのもたいへんですから、

$$A \times 5 = A \times (4 + 1) = A \times 4 + A$$

のように変形して、Aレジスタの内容を4倍したものに、さらに、4倍する前のAレジスタの内容

《第41図》プログラム1（実行不可能）

```
INC  A
LD   B, A
ADD  A, A
ADD  A, A
ADD  A, B
SUB  3
```

を加える事によって、5倍しています。

Aレジスタの内容を4倍するためには、加算命令を用いて、Aレジスタの内容を2倍し、その結果をさらに2倍しています。

最後に、

```
SUB  3
```

で、Aレジスタの内容から直接3を減じました。

これで、Aレジスタの内容に、1を加えた後5倍しさらに3を減じたわけですが、計算途中でBレジスタも用いました。

ただし、このプログラムで計算を行う場合、計算過程のどこかで、数値が1バイトを超えて桁あふれを起こした時には、下位1バイトのみが有効となり、桁あふれはその場で即切り捨てられてしまいます。初めの数値をAレジスタに与える時に注意してください。

プログラム例の実行方法

本書をお読みのみなさんでマシン語を勉強中の方は本書以外にも多くの専門書や関係雑誌などを読んでおられる事と思います。

この本だけでも毎章4～5本程度のプログラム例を掲載していますから、それら全ての文献のプログラム例を合わせると、かなりの数になる事でしょう。

マイクロコンピュータを、理解するためには、このような短いプログラムを、数多く実際に実行させてみる事が重要だと思うのですが、残念ながら

《第42図》プログラム2（実行可能）

```
LD    A, (E 0 0 0 H)
INC   A
LD    B, A
ADD   A, A
ADD   A, A
ADD   A, B
SUB   3
LD    (E 0 0 0 H), A
JP    5 C 6 6 H
```


全てのプログラム例が即実行できるかと申しますと、そうではないのです。

なぜならば、まず第一に各機種間の互換性の問題があります。Z80と、6800の様に使用されているCPUが異なる場合にはもちろんの事ですが、例えば、TRS-80とMZ-80の様におなじZ80を使用しているても、ROMやBASICシステム内のサブルーチンを使用していたり、V-RAMのアドレスが各機種間で異なったりするために、同じマシン語のプログラム例が全ての機種で同じ様に実行できるわけでは無いのです。

このような理由から、マシン語を勉強するための文献、資料などを選択する場合には、御自分でお持ちの機種用として発表されているものか、できるだけ機種への依存性が少ないものを選ぶべきでしょう。

第二にこうしたプログラム例全てが実行する事を目的としているわけではありません。この点では、私自身も責任を感じているのですが、誌面の都合や説明のし易さから、全てのプログラム例を、アSEMBルリストやダンプリストを付けた上で掲載するわけにはいかないのです。ニーモニックのみのプログラム例の場合には、読者の皆さんに御自分でアSEMBルしていただかなければなりません。その場合にはアSEMBラ等を用いれば簡単にアSEMBルを行う事ができますが、できれば、ニーモニック→マシン語の対応表をひきながらのハンド・アSEMBルを行う事によって、よりマシン語を覚える事ができると思います。

また、プログラム例の中には実行結果を確認で

きないものも多くあります。なぜなら、マシン語で扱うレジスタの内容を確認したり、レジスタに値を代入したりする事が、できない事が多いのです。マシン語を中心として考えていた、ワンボード・マイコンの頃は、モニタにこのような機能が付属していたものですが、最近のパーソナル・コンピュータは、BASIC中心であり、PC-8001等も例外ではありません。

この様な場合には、実行結果をレジスタではなく、ダンプして確認できるメモリ上や、V-RAM上に移して、調べるのが良いでしょう。

本書の第4章で紹介した、レジスタ表示プログラム、“MINI BREAK POINTER”などのユーティリティ・プログラムを利用するものも一つの方法だと思います。

説明だけでは無責任だと思いますので、先程説明した第41図のプログラム例を、PC-8001で実行し、実行結果を確認できる様にしてみましょう。

幸いな事に？ このプログラムは、そのままPC-8001で実行し、結果を調べてみる事ができません。

まず、実行直後にAレジスタに代入するための数値を、どこかのアドレスに入れておいて、ロード命令によって、Aレジスタに代入します。そして、全ての演算が終了した時点で、Aレジスタに入っている結果を、ロード命令で、今度はAレジスタから先程のアドレスに移します。最後は、ジャンプ命令で、

JP 5C66H

《第43図》プログラム2のアSEMBルリスト

| アドレス | マシン語 | ニーモニック | コメント |
|------|----------|---------------|------------------|
| D000 | 3A 00 E0 | LD A, (E000H) | Aレジスタに引数を代入する。 |
| D003 | 3C | INC A | Aレジスタの内容に1を加える。 |
| D004 | 47 | LD B, A | Aレジスタの内容を5倍する。 |
| D005 | 87 | ADD A, A | |
| D006 | 87 | ADD A, A | |
| D007 | 80 | ADD A, B | |
| D008 | D6 03 | SUB 3 | Aレジスタの内容から3を減じる。 |
| D00A | 32 00 E0 | LD (E000H), A | E000H番地に結果を入れる。 |
| D00D | C3 66 5C | JP 5C66H | マシン語モニタにジャンプする。 |

の様に、マシン語モニタにジャンプします。

Aレジスタへの引数を与え、結果を収納するのには、E000H番地を用い、結果は、マシン語モニタのDコマンドによって、

DE000, E000_{CR}

のようにして、調べる事ができます。

以上の変更を行ったプログラムが、第42図のプログラムで、さらにアセンブルしたものが第43図になります。プログラムは、D000H番地に置きましたが、どのアドレスに置いてもモニタのGコマンドで実行できます。

なお、当然の事ですが、E000H番地には、

16進数で引数を与え、結果も16進数で入ります。例えば、02HをE000H番地に入れてプログラムを実行させると、0CHが結果として得られるはずです。

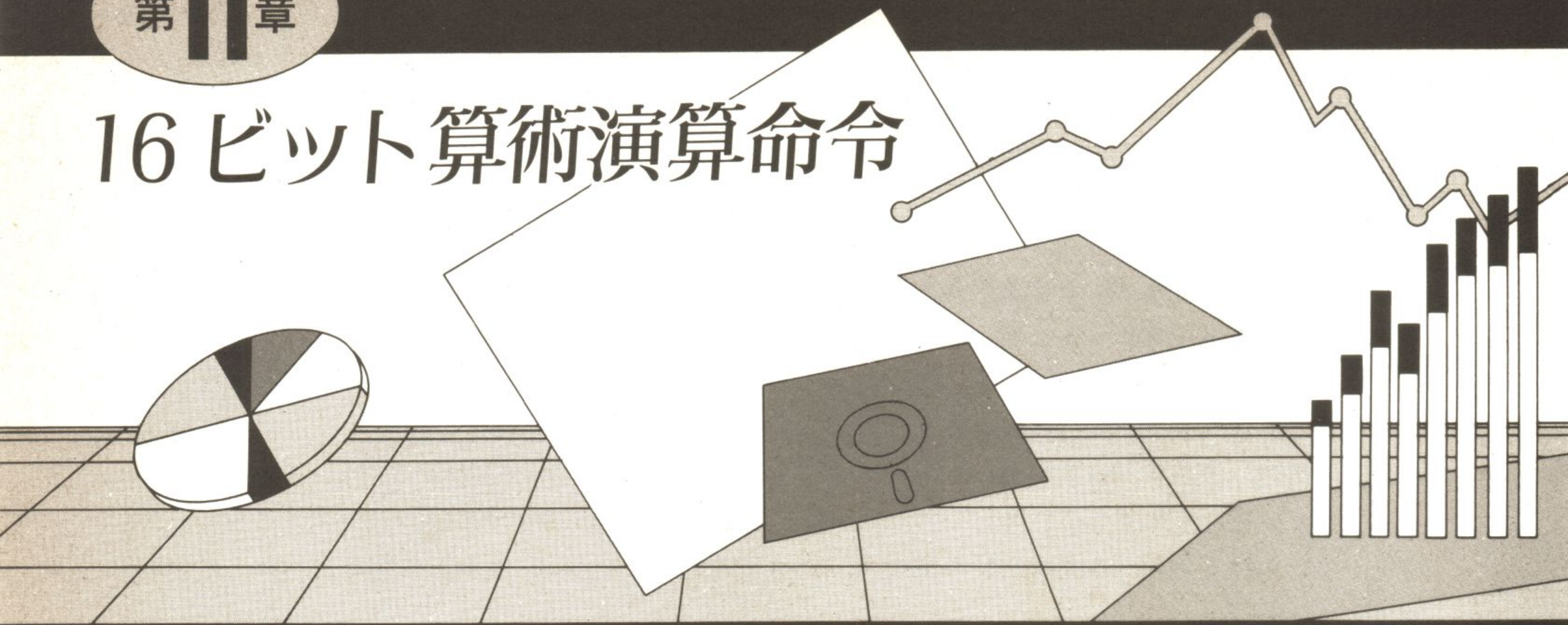
まとめ

本章では、8ビットのインクリメント命令とデクリメント命令を紹介しました。次章では、いよいよ16ビットの演算命令を行います。後は条件ジャンプさえ理解すれば、かなり楽しい事ができるはずですから、もう少しの間ガンバッテください。

《第44図》8ビット算術論理演算

| × | A | B | C | D | E | H | L | (HL) | (IX +d) | (IY +d) | n |
|----------|----|----|----|----|----|----|----|------|---------------|---------------|---------|
| ADD A, × | 87 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | DD 86 d | FD 86 d | C6 n |
| ADC A, × | 8F | 88 | 89 | 8A | 8B | 8C | 8D | 8E | DD 8E d | FD 8E d | CE n |
| SUB × | 97 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | DD 96 d | FD 96 d | D6 n |
| SBC A, × | 9F | 98 | 99 | 9A | 9B | 9C | 9D | 9E | DD 9E d | FD 9E d | DE n |
| AND × | A7 | A0 | A1 | A2 | AB | A4 | A5 | A6 | DD A6 d | FD A6 d | E6 n |
| XOR × | AF | A8 | A9 | AA | A3 | AC | AD | AE | DD AE d | FD AE d | EE n |
| OR × | B7 | B0 | B1 | B2 | B3 | B4 | B5 | B6 | DD B6 d | FD B6 d | F6 n |
| CP × | BF | B8 | B9 | BA | BB | BC | BD | BE | DD BE d | FD BE d | FE n |
| INC × | 3C | 04 | 0C | 14 | 1C | 24 | 2C | 34 | DD 34 d | FD 34 d | |
| DEC × | 3D | 05 | 0D | 15 | 1D | 25 | 2D | 35 | DD 35 d | FD 35 d | |

16ビット算術演算命令



16ビット算術演算命令

本章で紹介するのは16ビットのレジスタ間で算術演算を行うための16ビット算術演算命令です。

命令自体は、全て8ビット算術演算命令で説明したものばかりで、それらの16ビット版といったところですから、それほどわかりづらい事もないでしょう。

ただし命令自体は同じでもフラグの変化が8ビット版の算術演算命令とは異なりますので注意しなければなりません。

8ビットの算術演算命令ではほとんどのフラグに影響を与えていましたが、今回の16ビット算術演算命令では比較的影響を受けないフラグが多いのです。特に16ビットのインクリメント命令、デクリメント命令などは全くフラグを変化させません。

これは、8ビットの方が演算後の条件判断をより意識しているのに対して、16ビットの方はメモリ上のデータ扱う場合のポインタなどとして用いる事を第一に考えているためでしょう。

今のところは、条件判断の命令を説明していませんから問題はないと思いますが、条件ジャンプ命令などを頻繁に用いる様になると重要な問題になるでしょう。

私も前に、月刊マイコン誌上の「オセロ大会」に参加すべくオセロ・ゲームの探索ルーチンを制

作中に、16ビットのデクリメント命令後にゼロ・フラグ(Z)が変化すると信じ込んだ条件ジャンプ命令を行ったバグが、どうしてもわからずに制作を中断し、気が付いた時には、制作意欲が無くなっていた経験があります。その頃は、アセンブラなど未だ無く、全てハンド・アSEMBルでしたので、制作意欲はプログラミング上かなり重要な位置を占め、そのプログラムは未完成に終わりましたが、読者の皆さんもこのような事のないように、フラグの変化には気を付けてください。

使い易いアセンブラやデバッキング・ツールなどが出回り、マシン語のプログラムも組み易くはなってきましたが、それでも根本的なアルゴリズムの間違ひだけは人間の注意によって防がなければなりません。結局アセンブラが面倒を見てくれるのは、

Syntax error (記述ミス) だけなのです。

◎HLレジスタ対の内容に16ビットのレジスタ(対)の内容を加える命令

8ビットの加算命令では、Aレジスタとレジスタの内容を加えて、結果は必ずAレジスタに収納されましたが、16ビットの加算命令ではAレジスタのかわりに、HLレジスタ対、IXレジスタ、IYレジスタを用いる事ができます。

HLレジスタ対を用いる場合には、HLレジスタ対と、BCレジスタ対、DEレジスタ対、HL

レジスタ対、スタック・ポインタ (SP) の各レジスタの間で和を求める事ができます。

スタック・ポインタ (SP) との間で、加算を行う事ができるのも、16ビットの演算命令がアドレス制御を意識しているためでしょう。この命令はシステム関係のプログラム、特にデバッカなどスタック・ポインタの値を調べる必要がある場合には大へん便利で、

```
LD HL, 0000H
```

```
ADD HL, SP
```

を行う事で、スタック・ポインタ (SP) の内容がHLレジスタ対に移されます。

また、HLレジスタ対の内容を2倍する場合などに

```
ADD HL, HL
```

を用いる事もできます。

いずれにしてもHLレジスタ対は、アドレス指定の中心となるものですから、このレジスタ対を中心として16ビットの演算が行える事によって、80系CPUの弱点であるアドレッシング・モードの貧弱さを十分にカバーしていると思います。

HLレジスタ対を対象とした16ビットの加算命令は1バイト命令ですから省メモリの点からも安心して使える命令と言えるでしょう。

◎IXレジスタの内容に16ビットの

レジスタ(対)の内容を加える命令

前述した命令の対象がHLレジスタ対からIXレジスタに移ったものと考えれば良いでしょう。

ただし、マシン語にアセンブルするとHLレジスタ対を対象とする場合より、1バイト目にDDHが付いて、メモリを1バイト多く必要とします。

◎IYレジスタの内容に16ビットの

レジスタ(対)の内容を加える命令

命令の対象がIYレジスタに移ったものです。

◎HLレジスタ対から16ビットの

レジスタ(対)の内容を減じる命令

HLレジスタ対から他のレジスタ(対)の内容を減じた差をHLレジスタ対に求めるための命令ですが、同時にキャリー・フラグ (CY) も減じてしまいますから、例えばHLレジスタ対からD

Eレジスタ対の内容を減じる場合には、

```
SUB HL, DE
```

ではなく

```
SBC HL, DE
```

となります。

キャリー・フラグ (CY) を含めない普通の減算命令が用意されていないのは、多少不便な点もありますが、キャリー・フラグ (CY) を0にリセットしておいてから、この命令を実行すれば同じ事です。キャリー・フラグ (CY) を0にリセットするためにはいくつかの方法がありますが、本書の第9章でも紹介した8ビット論理演算命令を実行すれば良いので、他に影響を与えない

```
AND A
```

などを行い、キャリー・フラグを0にリセットしてから、SBC命令を実行すれば良いでしょう。

HLレジスタ対から、単にBCレジスタ対の内容を減じるためには、

```
AND A
```

```
SBC HL, BC
```

となります。

以上の例でもわかる様に、16ビットの減算命令というのは、単に16ビットの減算を行うためではなく、もう少し多ビットに渡る減算を行うために用意されているのです。

また、この命令では、HLレジスタ対からBCレジスタ対、DEレジスタ対、HLレジスタ対、スタック・ポインタ (SP) の内容を減じる事ができ、全て2バイト命令となります。

◎16ビットのレジスタ(対)に1を加える命令

16ビットのレジスタ(対)用のインクリメント命令で、指定したレジスタ(対)の値に1を加える事ができます。

実際に使用する場合には、ループ(くり返し)処理中などで、アドレス指定用のレジスタ対をインクリメントして行き、メモリ中のデータを扱う事が多く、ループ処理中でこそ、この命令の本領が発揮できるのではないのでしょうか？

この命令によってインクリメントできる16ビットのレジスタ(対)はBCレジスタ対、DEレジスタ対、HLレジスタ対、スタック・ポインタ(S

P), IXレジスタ, IYレジスタの6組で, 例えばHLレジスタ対に1を加えたい場合

INC HL

スタック・ポインタ (SP) に1を加えるには,

INC SP

となります。

もっとも, スタック・ポインタ (SP) をインクリメントする事はあまりないでしょう。

◎16ビットのレジスタ(対)から1を減じる命令

16ビットのレジスタ (対) 用のディクリメント命令で, 指定したレジスタ (対) の内容から1を減じる事ができます。

対象となるレジスタなどは全て, インクリメント命令と同じで, ニーモニックは8ビットのディクリメント命令と同じ様に, 例えばHLレジスタ対から1を減じる場合には,

DEC HL

と記述します。

16ビット算術演算命令のプログラム例

本章で説明した16ビット算術演算命令の使用例として, 32ビットに渡る数値の間で, 加算と減算を行うためのプログラムを紹介します。

現在のパーソナル・コンピュータは, PC-8001も含めて, ほとんどが8ビット並列処理のCPUを用いています。それに対して, 一般的な汎用コンピュータでは32ビットの数値を並列に処理できるようになっており, PC-8001のマシン語でも32ビットの演算に挑戦してみようと思い, この例題を考えました。

32ビットとは言っても恐れる事は無く, Z80の16ビット算術演算命令を用いれば二度で計算できます。8ビットの算術演算命令で16ビットの演算を行うのと, 理論的には同じ事です。

実際には, E000H~E003H番地と, E004H~E007H番地に, それぞれ32ビットに渡る数値を入れてプログラムを実行させると, 両者の和がE008H~E00BH番地に, 差がE00CH~E00FH番地に収納されるように

しました。

また, それぞれの数値は80系CPUによる演算の行い易さを考えてメモリの上位(数の多い方)から下位に向かって入れて行きます。

例をあげますと,

87654321H + 12345678H = 99999999H

87654321H - 12345678H = 7530E C A 9 H

を求めるためには, メモリ上には,

E000: 21 43 65 87

E004: 78 56 34 12

のようにセットして, プログラムを実行すると,

E008: 99 99 99 99

E00C: A9 EC 30 75

のように, 演算結果を得る事ができます。

人間から見るとかなりわかりにくい順番なのですがCPUにしてみれば, このならべ方が一番処理し易いのです。もちろんプログラム中でならべかえる事もできますが, プログラムが長くないように, ここではがまんしましょう。もっとも, 良く考えてみると, メモリの上位に, 数値の上位バイトを対応させる, 80系CPUの方が本当は正しいのかもしれないね?

16ビット算術演算命令が

よく理解できない方は

第8章へあとまどリ



プログラム例の説明

《第45図》 16ビット算術演算

(ニーモニック↔機械語対照表)

| × | BC | DE | HL | SP | IX | IY |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| ADD HL, × | 0 9 | 1 9 | 2 9 | 3 9 | | |
| ADD IX, × | DD 0 9 | DD 1 9 | | DD 3 9 | DD 2 9 | |
| ADD IY, × | FD 0 9 | FD 1 9 | | FD 3 9 | | FD 2 9 |
| ADC HL, × | ED 4 2 | ED 5 A | ED 6 A | ED 7 A | | |
| SBC HL, × | ED 4 2 | ED 5 2 | ED 6 2 | ED 7 2 | | |
| INC × | 0 3 | 1 3 | 2 3 | 3 3 | DD 2 3 | FD 2 3 |
| DEC × | 0 B | 1 B | 2 B | 3 B | DD 2 B | FD 2 B |

プログラムのアドレスは、D 0 0 0 H～D 0 3 2 H 番地です。実行する場合には、プログラム・リストのオブジェクト部分を打ち込んでくだされば、マシン語モニタの G (ジャンプ) コマンドで走らせる事ができます。当然ですが演算は一瞬で終わり、モニタにもどりますから、D (ダンプ) コマンドなどを用いて、E 0 0 8 H 番地以降の演算結果を知ることになります。

プログラムは、まず大きく、加算を行って和を求める部分と、減算によって差を求める部分に分ける事ができ、さらにそれぞれを、16ビットずつに分けて計算しています。

加算、減算共に、まず下位16ビットを普通に計算し、次に上位16ビットをキャリー・フラグ (C

《第46図》 16ビット算術演算命令プログラム例

| アドレス | オブジェクト | ニーモニック | コメント |
|---------|----------------|--------------------|----------------|
| D 0 0 0 | 2 A 0 0 E 0 | LD HL, (E 0 0 0 H) | 下位16ビットを加算 |
| D 0 0 3 | ED 5 B 0 4 E 0 | LD DE, (E 0 0 4 H) | |
| D 0 0 7 | 1 9 | ADD HL, DE | |
| D 0 0 8 | 2 2 0 8 E 0 | LD (E 0 0 8 H), HL | |
| D 0 0 B | 2 A 0 2 E 0 | LD HL, (E 0 0 2 H) | 上位16ビットを加算 |
| D 0 0 E | ED 5 B 0 6 E 0 | LD DE, (E 0 0 6 H) | |
| D 0 1 2 | ED 5 A | ADC HL, DE | |
| D 0 1 4 | 2 2 0 A E 0 | LD (E 0 0 A H), HL | |
| D 0 1 7 | 2 A 0 0 E 0 | LD HL, (E 0 0 0 H) | 下位16ビットを減算 |
| D 0 1 A | ED 5 B 0 4 E 0 | LD DE, (E 0 0 4 H) | |
| D 0 1 E | A 7 | AND A | |
| D 0 1 F | ED 5 2 | SBC HL, DE | |
| D 0 2 1 | 2 2 0 C E 0 | LD (E 0 0 C H), HL | |
| D 0 2 4 | 2 A 0 2 E 0 | LD HL, (E 0 0 2 H) | 上位16ビットを減算 |
| D 0 2 7 | ED 5 B 0 6 E 0 | LD DE, (E 0 0 6 H) | |
| D 0 2 B | ED 5 2 | SBC HL, DE | |
| D 0 2 D | 2 2 0 E E 0 | LD (E 0 0 E H), HL | |
| D 0 3 0 | C 3 6 6 5 C | JP 5 C 6 6 H | マシン語モニタにジャンプする |

Y)を含めた演算命令で計算しており、常にHLレジスタ対とDEレジスタ対に数値を持ってきては、両者の間で演算命令を行い、結果をメモリ上に収納しています。ただ、先程述べた様にキャリー・フラグ(CY)を含めない普通の16ビット減算命令はありませんので、関係のない論理演算命令

AND A

を行って、キャリー・フラグ(CY)を0にリセットしておいてから、キャリー・フラグ(CY)を含む

SBC HL, DE

を用いて、かわりにしました。

演算が終了すると、

JP 5C66H

で、マシン語モニタにジャンプします。

以上のように、非常に単純なプログラム例です

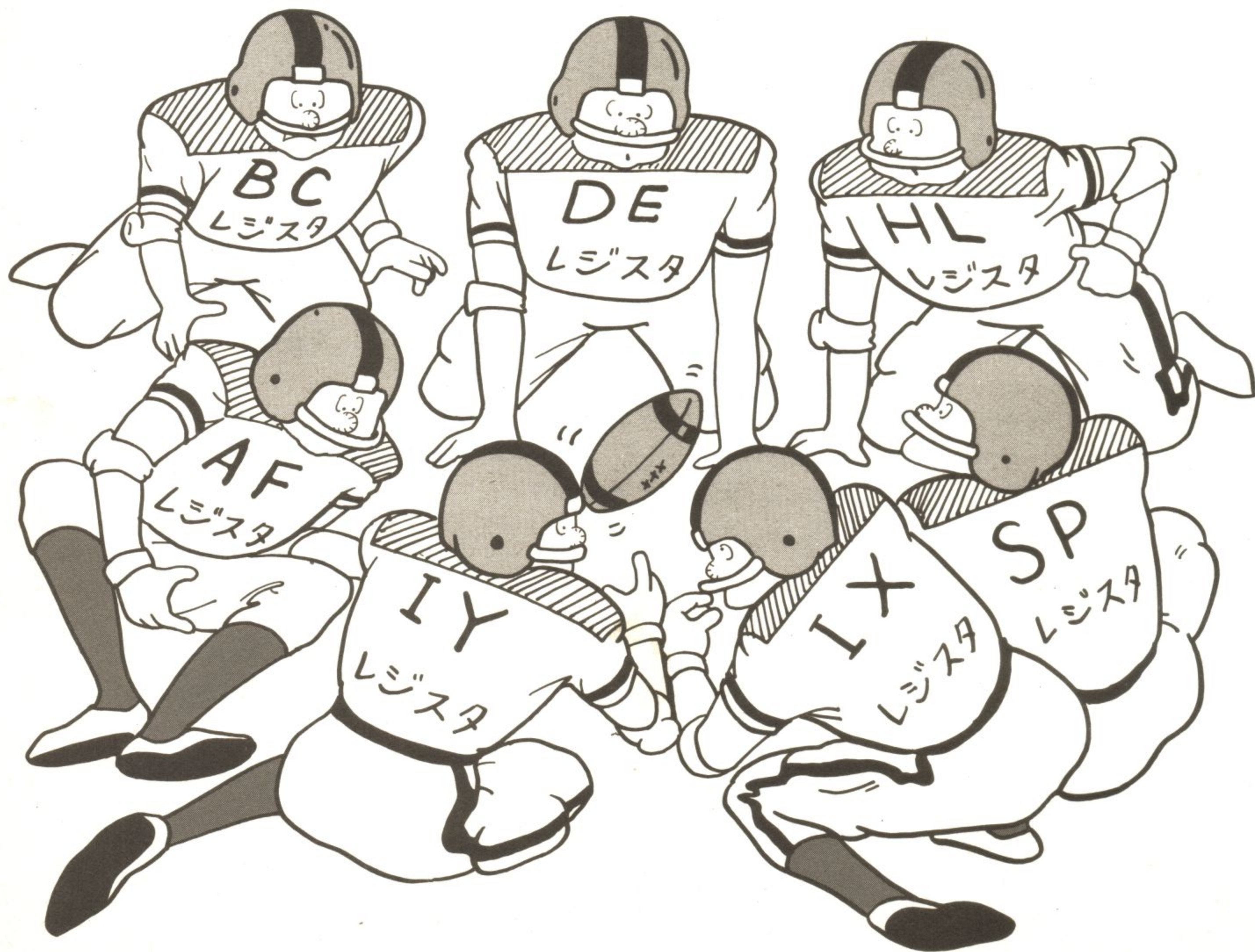
から理解しづらい部分もないでしょう。又、このプログラム例では、インクリメント命令やデクリメント命令を使いませんでした。それらの命令は今後続々と使用しますので楽しみにしてください。

まとめ

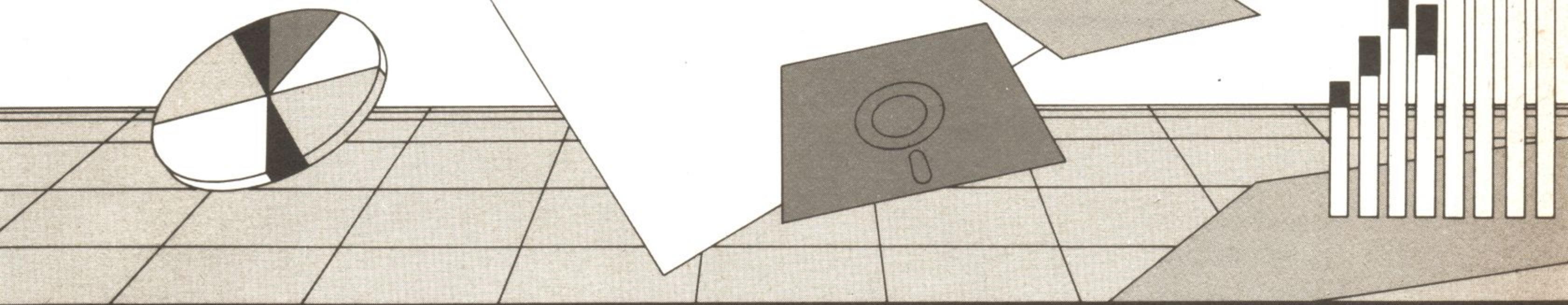
16ビット算術演算命令は、重要な命令の割に説明する事項が少なかったため、多少もの足りなさを感じた皆さんもいらっしゃる事でしょう。

次の章では、いよいよ条件ジャンプ命令の登場です。これで少しは、複雑なプログラム例が出せるのではないかと、私自身も期待しています。

16ビットロード命令 算術演算命令に 活躍するレジスタ群



ジャンプ命令



8ビット算術・論理演算命令補足

第8章から第10章にわたり8ビットの算術・論理演算命令を紹介しましたが一つ重要な命令を忘れていました。

条件ジャンプを行う前にぜひ、知っておいて欲しい命令ですから、ここで説明しておきましょう。

◎Aレジスタとレジスタの

内容を比較する命令

ところで、マシン語で、数値を比較するには、どうすれば良いのでしょうか？ 普通に考えた場合には、減算命令を行った直後に、フラグの変化を調べれば良いのです。例えばAレジスタとBレジスタの内容を比較するためには、

```
SUB B
```

を行う事になりますが、これではAレジスタに差が入ってしまい、困る場合が生じます。そこで、他のレジスタを、Aレジスタの一時退避用に使って、

```
LD C, A
```

```
SUB B
```

```
LD A, C
```

の様に、Aレジスタの内容をもとにもどしておかなければなりません。そこで、差を求めない減算命令が必要になってくるのですが、それが比較命令です。

上記の様にAレジスタとBレジスタの内容を比

較したいのであれば、

```
CP B
```

となります。CPは、ComPareの略で、この命令実行後には、どのレジスタも変化しないのですが、例外的に、フラグのみが、減算命令を行ったのと同じ様に变化するのです。つまり、AレジスタとBレジスタの内容が等しければ、ゼロ・フラグ(Z)が1にセットされ、Bレジスタの内容の方が大きければ、キャリー・フラグ(CY)が1にセットされる事になります。

◎Aレジスタと直接与える数値を比較する命令

Aレジスタと数値を比較する命令で、比較命令は、この形で使われる事が多いようです。

例えば、Aレジスタの内容が、“●”のキャラクター・コードかどうかを調べるためには、

```
CP ECH
```

を行い、ゼロ・フラグが1にセットされれば“●”である事がわかります。

アセンブルする場合には、2バイト目に比較する数値を直接割り当てます。

◎Aレジスタと指定メモリの内容を比較する命令

Aレジスタと、メモリ上の数値を順序よく比較して行く場合などに用いる命令で、メモリのアドレス指定にはおなじみの、HLレジスタ対、IXレジスタ、IYレジスタを使う事ができます。

簡単な命令ですが、使用例として、Aレジスタ

とE000H番地の内容を比較する方法をあげておきます。

```
LD    HL, E000H
CP    (HL)
```

比較結果は当然フラグの変化となって表われます。

ジャンプ命令

これから紹介する、Z80の命令は、ジャンプ命令です。

無条件ジャンプ命令は、N—BASICでは、
GOTO 〈行番号〉

にあたりますが、条件ジャンプ命令は、

```
IF 〈条件式〉 THEN 〈行番号〉
(GOTO)
```

にあたります、両者共に大変重要な命令です。

また、ジャンプ命令には、ジャンプ先のアドレスを、絶対的なアドレスで指定する絶対ジャンプ命令と、相対的に指定する相対ジャンプ命令がありますが、本章では、絶対ジャンプ命令を中心に説明を行い、相対ジャンプ命令をアセンブルする場合のアドレス計算については、次の章に説明を延ばしたいと思います。

絶対ジャンプ命令のニーモニックは、

```
JP (JumP)
```

の後に、オペランドを置きますが、相対ジャンプ命令では、

```
JR (Jump Relative)
```

となりますので、覚えておいてください。

◎無条件にジャンプを行う無条件ジャンプ命令

無条件ジャンプ命令とは、普通のジャンプ命令のことで、今までにもプログラム例の最後で、

```
JP 5C66H
```

として使ってきました。

この例では、5C66H番地、つまりPC—8001のROM内に有る、マシン語モニタの開始アドレスにジャンプさせていますが、

```
JP 0081H
```

を行えば、N—BASICにもどす事ができます。

ところで、皆さんはジャンプ命令を実行する場

合にCPUがどのような動作を行うか想像が付きませんか？ 何か複雑な作業を行っているかの様に思えますが実に簡単、プログラム・カウンタ(PC)に、ジャンプ先のアドレスを代入しているだけなのです。

ですから、マシン語のジャンプ命令は、BASICなどのGOTO文と異なり、いくら多用しても実行速度にはほとんど影響を与えません。プログラムが見つらくなる事さえ気にしなければ、安心してジャンプ命令を使用する事ができます。

マシン語にアセンブルする場合には、3バイト命令となり、1バイト目はC3で、2バイト目、3バイト目には、ジャンプ先のアドレスを上位、下位8ビットごとに分けて割りあてます。先程の、

```
JP 5C66H
```

であれば、

```
C3・66・5C
```

と、アセンブルしてください。

また、無条件ジャンプ命令は、相対ジャンプ命令を用いる事もできます。

◎ゼロ・フラグ(Z)が0の時にジャンプする命令

ジャンプ命令の中では、最も多く用いられる命令でしょう。一定回数のくり返し(ループ)処理を行う場合にはほとんどの場合に、このジャンプ命令を用いる事になります。

ところで読者の皆さんは、ループ処理について御存知でしょうか？ マイコン用語を見なれている方や、BASICなどの言語によるプログラミングの経験がある方は別として一般にはあまり使われない言葉ではないでしょうか？

第5章では、ロード命令だけを使って、直接ビデオ・ラムにキャラクタ・コードを送り込み、短いメッセージを、CRT画面上に表示させてみましたが、もしゲームの説明などで、1000～2000文字程度のメッセージを何種類も表示しなければならない場合には、どうすれば良いでしょうか？

第5章と同じ様な方式で、全ての文字をアドレスを指定しながら、V—RAMに送っていたのでは、長大なメッセージを表示するプログラムを組むために、32KバイトのRAM空間も、プログ

ラマの気力も尽きてしまう事は确实です。

そこで、同じような単純作業を、数回くり返すために、プログラムの流れを輪の様にすることが必要となりますが、これをループ処理と言います(第47図)。

しかも、永久的に同じ作業をくり返すことは、まずありませんから、一定回数だけくり返した後は、輪の中からぬけ出す必要があります(第48図)。

この、一定回数行うくり返し処理を、条件ジャンプ命令を用いて表わしてみましよう。合わせて、N-BASICのプログラムも掲載しておきますので、両者を比較してみてください(第49図)。

この例では、くり返し(ループ)回数を5回にしてあり、又、回数を数えるためのカウンタとして、Aレジスタを用いています。

Aレジスタの初期値は5ですが、

DEC A

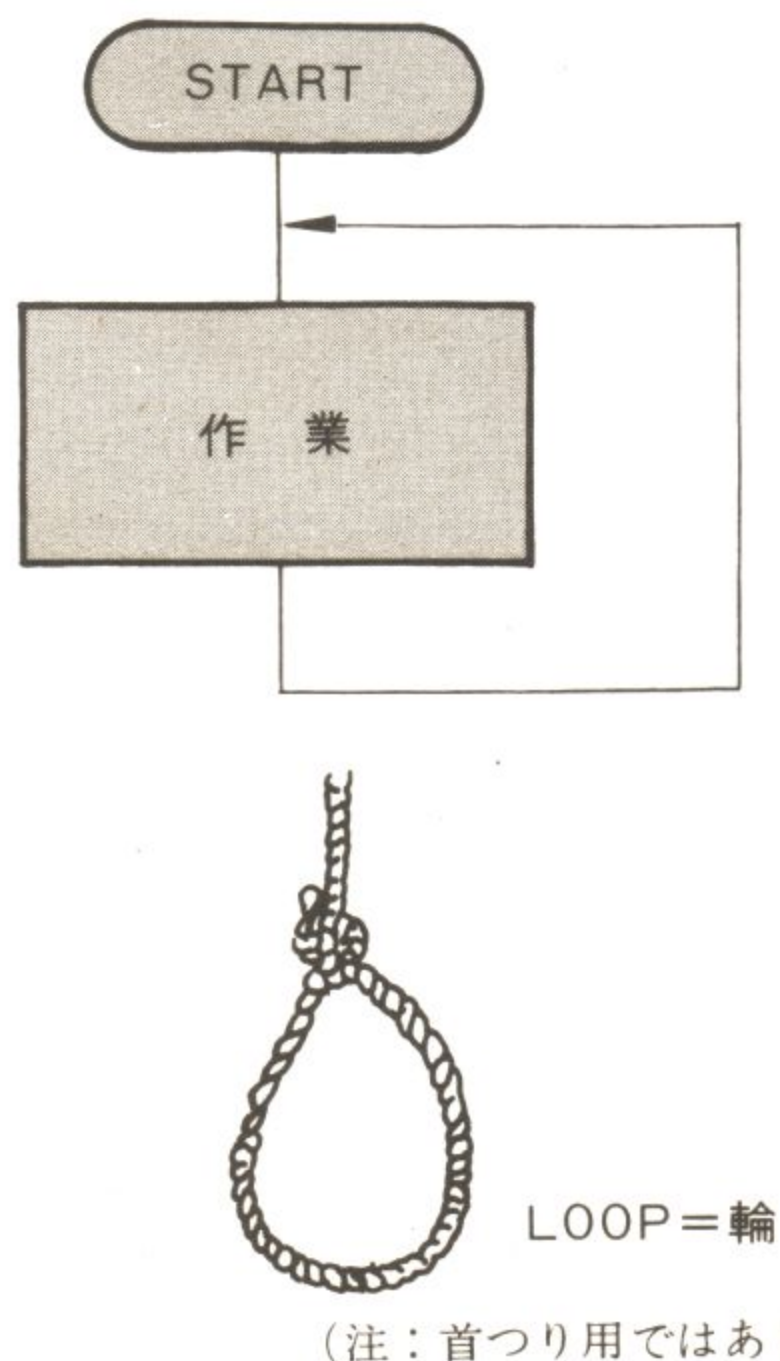
を行うたびに1ずつ減少して行き、5回めには、0になるために、ゼロ・フラグ(Z)が1にセットされて、LOOPの部分に、ジャンプしなくなります。

ここで使われている、

JP NZ, LOOP

という命令が、ゼロ・フラグ(Z)が0の時にジャンプを行う条件ジャンプ命令で、NZは、No

《第47図》無条件ジャンプによるループ処理



t Zeroの略です。

LOOPと言うのは、第1章で述べた、ラベル(LABEL)と呼ばれているもので、実際にプログラムを実行させる場合には、アドレスが入るのですが、コーディングする(プログラムを書く)段階では、分かり易いように、英文字と数字などの組み合わせを書いておくものです。

また、この命令もJPをJRにかえて相対ジャンプ命令を使用する事ができます。

◎ゼロ・フラグ(Z)が1の時にジャンプする命令

ゼロ・フラグ(Z)の内容が1にセットされている場合に、ジャンプを行う命令です。ニーモニックでは、

JP Z, ADR

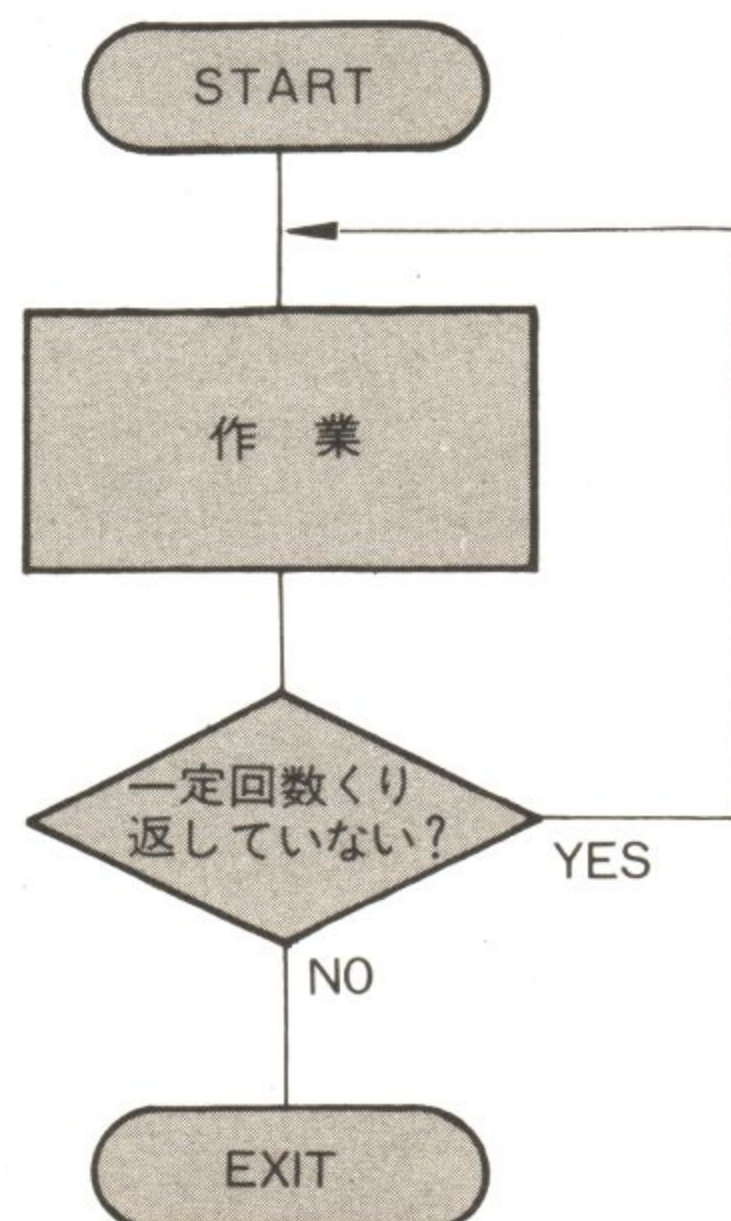
のようになります。ADRの部分は、実際にはジャンプ先のアドレスを入れます。

先程行った、第49図の5回のくり返し(ループ)作業を今度は、今説明している命令を用いて表わしますので、両者を比較してください(第50図)。

また、この命令は、相対ジャンプ命令を使用する事ができます。

《第48図》条件ジャンプによる

一定回数のループ処理



《第49図》 5回のくり返し作業（NZを使用）

| | | | | | |
|-------|-----|----------|----|----------|-----------------------|
| | LD | A, 05H | 10 | LET | A=5 |
| LOOP: | | 作業 | 20 | | 作業 |
| | | | 30 | | |
| | DEC | A | 40 | A=A-1:IF | A=0 THEN Z=1 ELSE Z=0 |
| | JP | NZ, LOOP | 50 | IF | Z=0 THEN 20 |
| | JP | 5C66H | 60 | END | |

◎キャリー・フラグ(CY)が0の時に

ジャンプする命令

キャリー・フラグ（CY）が0にリセットされている場合にジャンプを行う命令で、

JP NC, ADR

と書きます。NCは、Not Carryの略で、ADRは、何度も述べた様に、ラベルです。

この命令による、5回のくり返し（ループ）作業を示します（第51図）。

ゼロ・フラグ（Z）を用いて、ループ終了の判断を行っていた場合には、Aレジスタの内容が、

5 → 4 → 3 → 2 → 1 → 0

と変化していたのに対して、キャリーフラグ（CY）を用いた場合には、

0 → 1 → 2 → 3 → 4 → 5

と、増加して行き、5になった時点の、

CP 05H

で、キャリー・フラグ（CY）が0にリセットされてループからぬけ出す点に着目してください。

キャリー・フラグ（CY）はこのように、桁下りが生じている場合には、1にセットされていますが、そうでない場合には、0にリセットされます。

また、この命令は、相対ジャンプ命令を使用する事ができます。

◎キャリー・フラグ(CY)が1の時に

ジャンプする命令

キャリー・フラグ（CY）が1にセットされている場合にジャンプする命令で、

《第50図》 5回のくり返し作業（Zを使用）

| | | | | | |
|-------|-----|----------|----|----------|-----------------------|
| | LD | A, 05H | 10 | LET | A=5 |
| LOOP: | | 作業 | 20 | | 作業 |
| | | | 30 | | |
| | DEC | A | 40 | A=A-1:IF | A=0 THEN Z=1 ELSE Z=0 |
| | JP | Z, 5C66H | 50 | IF | Z=1 THEN END |
| | JP | LOOP | 60 | GOTO | 20 |

《第51図》 5回のくり返し作業（NCを使用）

| | | | | | |
|-------|-----|-----------|----|-------|---------------------------|
| | LD | A, 00H | 10 | LET | A=0 |
| LOOP: | | 作業 | 20 | | 作業 |
| | | | 30 | | |
| | INC | A | 40 | A=A+1 | |
| | CP | 05H | 50 | IF | A-5<0 THEN CY=1 ELSE CY=0 |
| | JP | NC, 5C66H | 60 | IF | CY=0 THEN END |
| | JP | LOOP | 70 | GOTO | 20 |

《第52図》 5回のくり返し作業 (Cを使用)

| | | | | | |
|-------|-----|---------|----|------------------------------|-----|
| | LD | A, 00H | 10 | LET | A=0 |
| LOOP: | | 作業 | 20 | | 作業 |
| | | | 30 | | |
| | INC | A | 40 | A=A+1 | |
| | CP | 05H | 50 | IF A-5<0 THEN CY=1 ELSE CY=0 | |
| | JP | C, LOOP | 60 | IF CY=1 THEN 20 | |
| | JP | 5C66H | 70 | END | |

JP C, ADR

などと書きます。

同様に、5回のくり返し(ループ)作業を行うためのプログラムをあげておきますから、他のものと比較しておいてください(第52図)。

この命令までは全て相対ジャンプを使用できます。

◎サイン・フラグ(S)が0の時に

ジャンプする命令

サイン・フラグ(S)の内容が0の時、つまり直前に行った演算(フラグを変化させる)命令の実行結果が正(プラス)になった場合にジャンプする命令だと思って良いでしょう。

ニーモニックは、PlusのPを取って、

JP P, ADR

と、なります。

この命令は、キャリー・フラグ(CY)による条件ジャンプ命令で代用できる事が比較的多いので、相対ジャンプ命令は用意されていません。

◎サイン・フラグ(S)が1の時に

ジャンプする命令

サイン・フラグ(S)の内容が1の時、つまり直前に行った演算(フラグを変化させる)命令の実行結果が負(マイナス)になった場合にジャンプする命令だと思って良いでしょう。

ニーモニックは、MinusのMを取って、

JP M, ADR

と、なります。

◎パリティ/オーバーフロー・フラグ

(P/V)が0の時にジャンプする命令

パリティ/オーバーフロー・フラグ(P/V)が

0の時にジャンプする命令で、それがパリティ奇数か、オーバーフローによるものなのかは、直前に行った演算命令の種類によって決定されます。

ニーモニックは、Parity Odd(パリティ奇数)の略で、

JP PO, ADR

と、なりますが、ほとんど使われない命令でしょう。

◎パリティ/オーバーフロー・フラグ

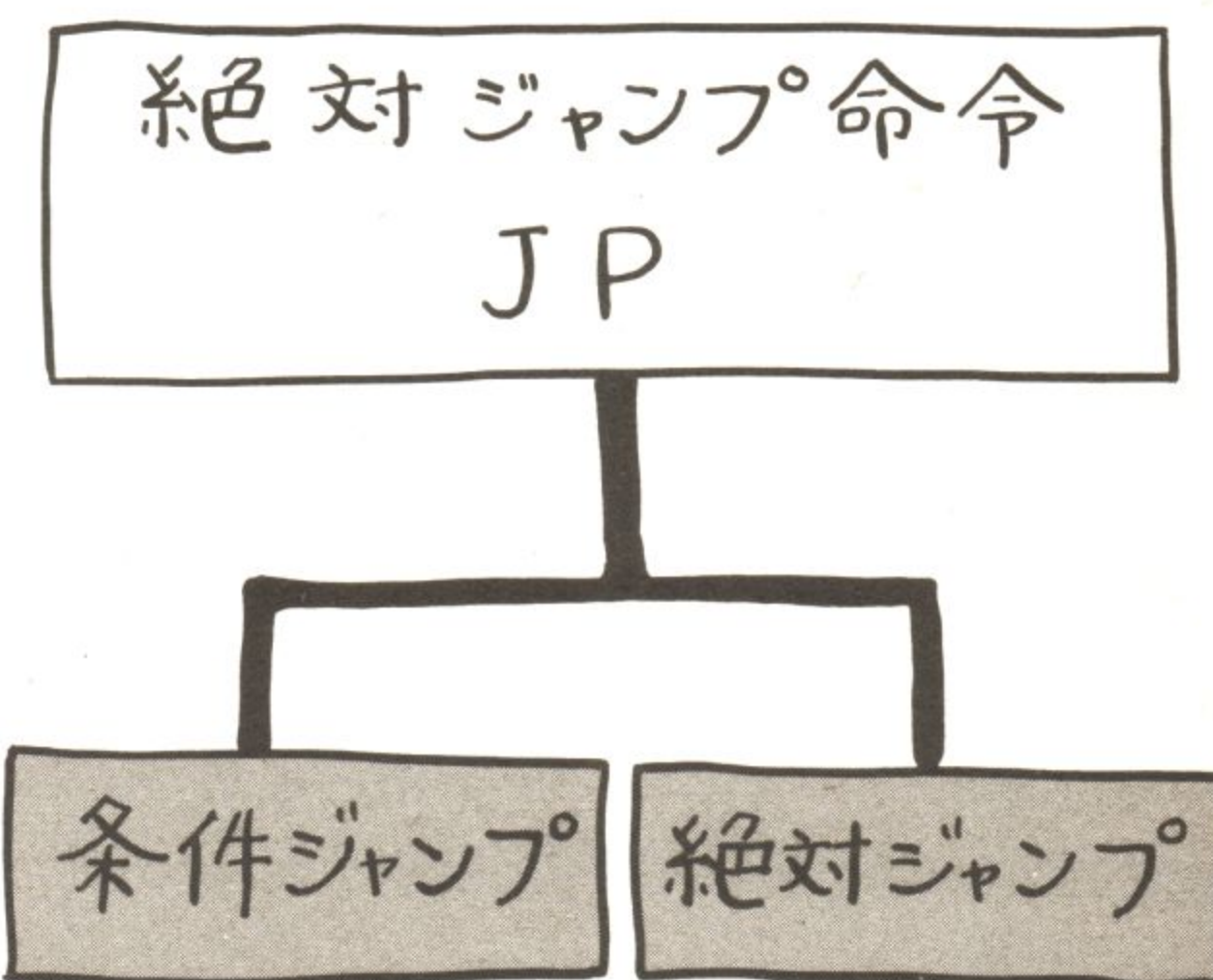
(P/V)が1の時にジャンプする命令

パリティ/オーバーフロー・フラグ(P/V)が1の時にジャンプする命令で、それが、パリティ偶数か、オーバーフローしない事によるものなのかは、直前に行った演算命令の種類によって決定されます。

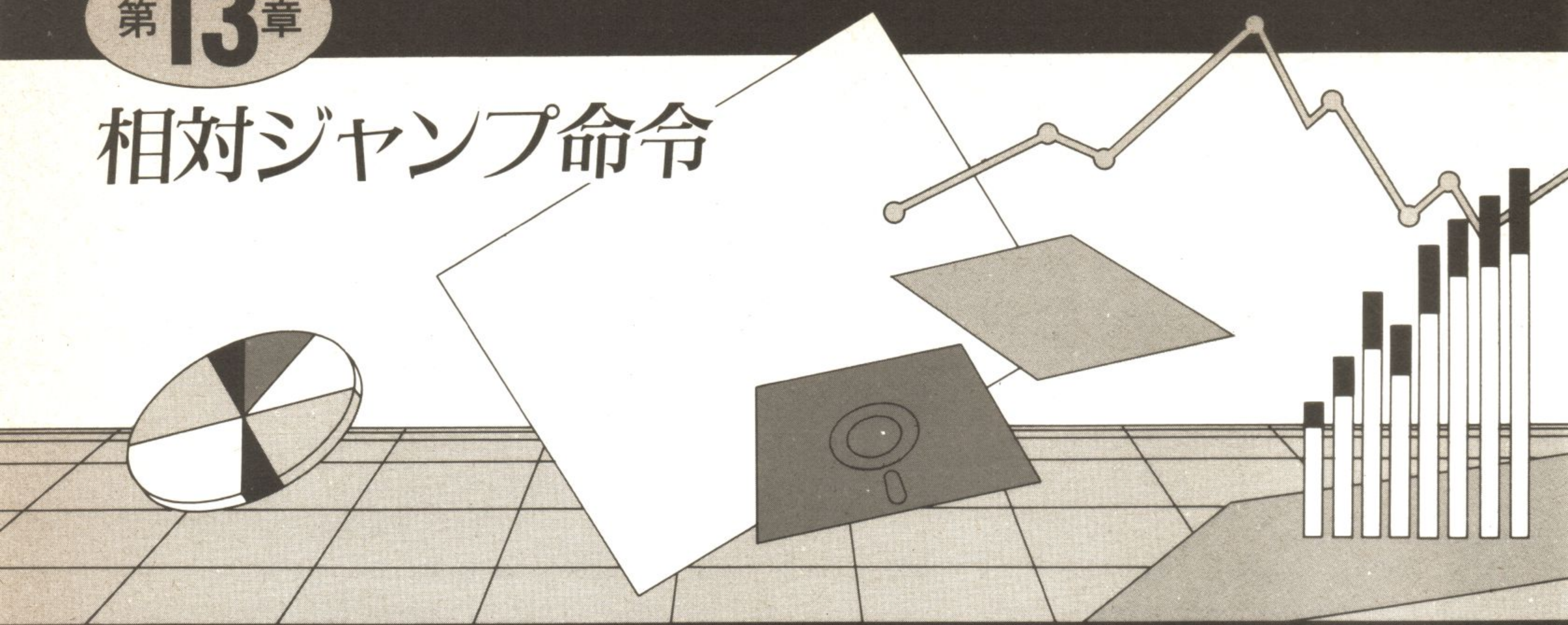
ニーモニックは、Parity Even(パリティ偶数)の略で、

JP PE, ADR

と、なりますが、ほとんど使われない命令でしょう。



相対ジャンプ命令



相対ジャンプ命令の 利用価値

Z-80 CPUには、8080 CPUのころから用いられてきた絶対ジャンプ命令が数多く用意されています。それらの命令は、マシン語命令の3バイトで成り立ち、2バイト目と3バイト目でジャンプ先アドレスを指定するのですから、使い方も大へん簡単で解り易い命令と言えましょう。

ところが、使い易さの反面、絶対ジャンプ命令ならではの欠点も存在するのです。

今、ある番地からに割り当てられている、マシン語のプログラムが用意されているとします。このプログラムを別のアドレス用に変換する事を、「リロケート」と言いますが、この、リロケートを行わなければならない機会は比較的多いのです。

例えば、Disk-BASICを用いているためにメモリの上方が使えない場合、同時に使いたいプログラムとアドレスが重なっている場合や他機種用のプログラムを別の機種に移植する場合など、リロケートする機会を上げれば、きりが無いほどです。

簡単なプログラムであれば、リロケートを行うためのプログラム、リロケータを用いる事によって比較的楽にリロケートを行う事ができますが、これを手作業で行うのは大へんです。

そこで、どこのアドレスに置いても実行する事ができる、リロケータブルなプログラムを組め

ば良いのですが、その時の障害になっているのが、絶対ジャンプ命令なのです。

御存知の様に絶対ジャンプ命令は、

「××××番地にジャンプする」

という様に、ジャンプ先のアドレスを絶対的に指定しなければなりませんから、リロケータブルなプログラムを組むには、絶対ジャンプ命令を用いる事はできません。

ところが、よほど短いテスト・プログラムなどを除いて、ジャンプ命令を使わない事は、有り得ませんから、

「××番地前にジャンプする」

「××番地後にジャンプする」

の様に、ジャンプ先のアドレスを相対的に指定する必要があります、相対ジャンプ命令を使用する価値が有るのです。

また、相対ジャンプ命令は全て2バイトの命令で、

1 バイト目が命令コード

2 バイト目がジャンプ先アドレスの指定用となっていますので、常に3バイトを使用する絶対ジャンプ命令に比較して省メモリ性にすぐれています。

以上の様に、使い勝手では、絶対ジャンプ命令の方が良いのですが、リロケータブルなプログラムを組める事と、省メモリのために、相対ジャンプ命令が使用されるのです。

相対ジャンプ命令の アドレス指定

前述した様に相対ジャンプ命令では、1バイトでジャンプ先のアドレスを指定します。

絶対ジャンプ命令では、2～3バイト目の内容がジャンプ先のアドレスとして直接、プログラム・カウンタ（PC）に入れられたため、ジャンプを行いました。相対ジャンプ命令では、2バイト目の内容が、符号付きの数値として、プログラム・カウンタ（PC）に加えられる事によって、ジャンプが実行されるのです。

本書の第8章にも表を掲載したように、1バイトでは、-128～+127までの数値しか表わす事ができませんから、当然ジャンプ先には制限があります。この範囲を越えてジャンプを行いたい時には、絶対ジャンプ命令を使用するか、途中で中継点をつかったプログラムを構成する事が必要です。

さて、相対ジャンプを使う上で一番問題になるのが、2バイト目で指定するためのジャンプ先アドレスを計算する方法でしょう。

ほとんどの解説書では、計算によって、求める事を勧めている様ですが、私が勧めるのは、1バイトずつ数えて行くという、非常に原始的な方法です。

J **P** E 000H 絶対ジャンプ

J **R** E 000H 相対ジャンプ

JPとJRの
違いを
よく理解
しましょう



ハンド・アセンブルするのは、ほとんどが相対ジャンプ命令を、余り多くは使っていない短いプログラムですから、1バイトずつ数えていっても、たかが知れています。最高でも、128バイト数えるだけです。計算している間に数えた方が早いと思います。

その際に注意しなければならないのは、全くジャンプを行わない時に、00Hを与える事で、自分自身のアドレスにジャンプするには、FEHとなります。

相対ジャンプの使える ジャンプ命令

前章で紹介した絶対ジャンプ命令のほとんどに、同じ用途に用いるための相対ジャンプ命令が用意されており、それぞれのニーモニックの、

JP

を

JR

に変える事で、相対ジャンプ命令である事を表わす事ができます。

例えば、絶対ジャンプ命令の、

JP ADDRESS

は、

JR ADDRESS

となります。この

JR

は、

Jump Relative

の略で、文字どおり

相対ジャンプ

の事です。

それではここで、相対ジャンプ命令を全て紹介しましょう。ただし、ジャンプ先は

ADDRESS

とし、特殊な命令は後で説明するために省いておきます。

JR ADDRESS

JR Z, ADDRESS

JR NZ, ADDRESS

JR C, ADDRESS

J R NC, ADDRESS

以上です。

特殊なジャンプ命令

今まで説明したジャンプ命令とは少し変わったジャンプ命令を、2種類ほど紹介しましょう。

両者共、特殊な命令で、別の命令を使っても代用する事ができるのですが、知っているると非常に役立つ命令ですから、それぞれにプログラム例をあげておきました。

BASICには無い、マシン語独特の感覚を持った命令です。

◎ジャンプ先のアドレスを

16ビットのレジスタ（対）で指定する命令

通常のジャンプ命令では、命令の2～3バイト目でジャンプ先のアドレスを指定するのですが、この命令ではレジスタ（対）の値がジャンプ先のアドレスを示します。

ほとんどのジャンプ命令は、ジャンプ先が常に一定ですが、例えば、BASICの

ON～GOTO～

のように、演算結果やキー入力などによってジャンプ先を変えて行く場合には、レジスタ（対）の値でアドレス指定を行う事が必要になります。

アドレス指定には、

HLレジスタ対

IXレジスタ

IYレジスタ

を用いる事ができ、それぞれのニーモニックは、

J P (HL)

J P (IX)

J P (IY)

と書きます。

それでは、演算命令の復習もかねて、HLレジスタ対の値による条件分岐を行ってみましょう。プログラム例では、HLレジスタ対の値による計算型ジャンプを行ってみました。

第53図に、HLレジスタ対の値とジャンプ先アドレスの対応と、実際のプログラム例を示します。

HLレジスタ対の内容を16倍（10H倍）して、E000Hを加えた後に、

J P (HL)

を行っているだけですが、この命令の最も基本的な使い方だけは、わかっていただけたと思います。

この命令を使わない場合は、同じ事を行うのに、プログラム自身を書き換えたり、スタックを巧みに利用しなければならず、大へん見づらいプログラムになるのですが、この命令を使用することによって、かなりプログラミングが楽になっています。

参考のために、スタックを利用した同じプログラムをあげておきます。まだ

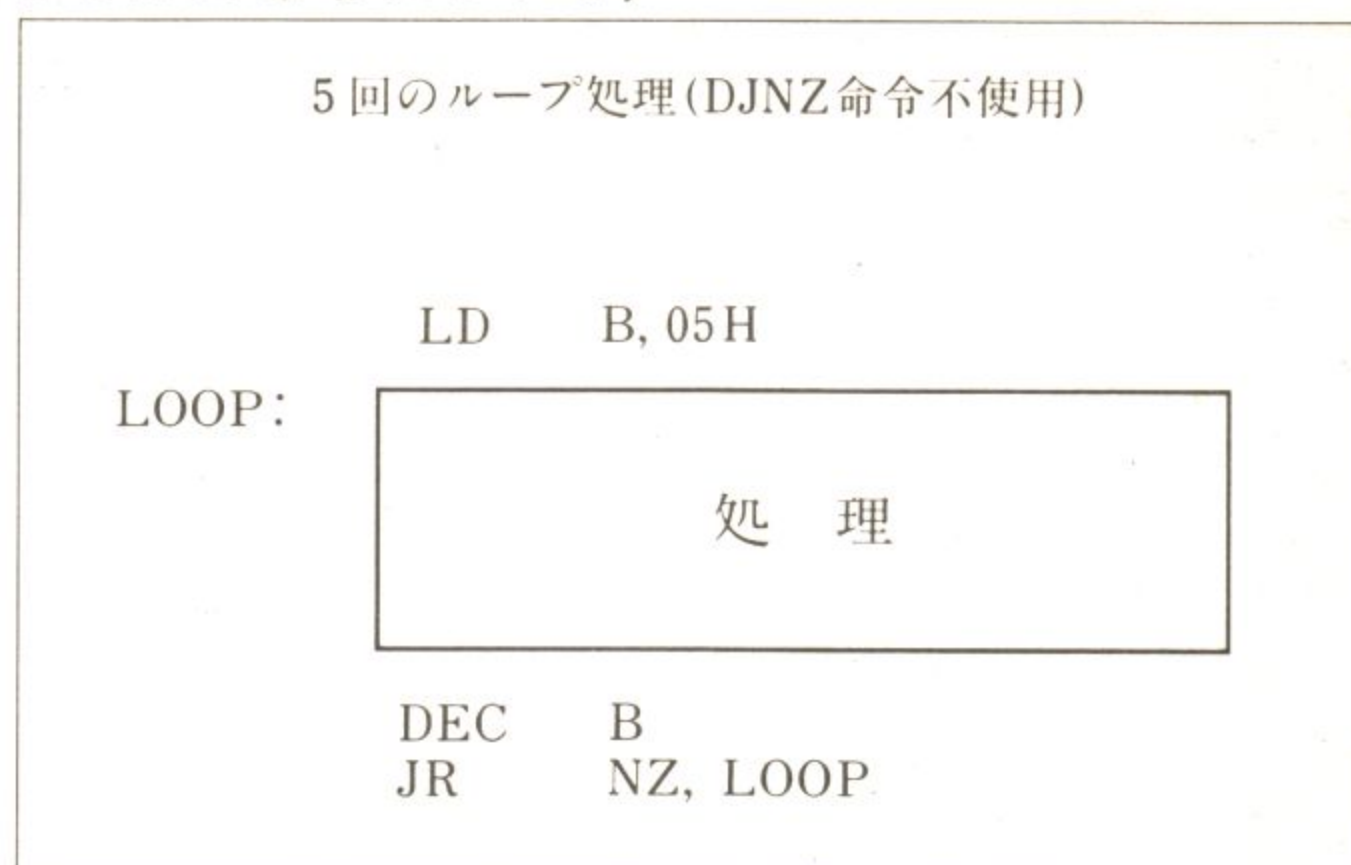
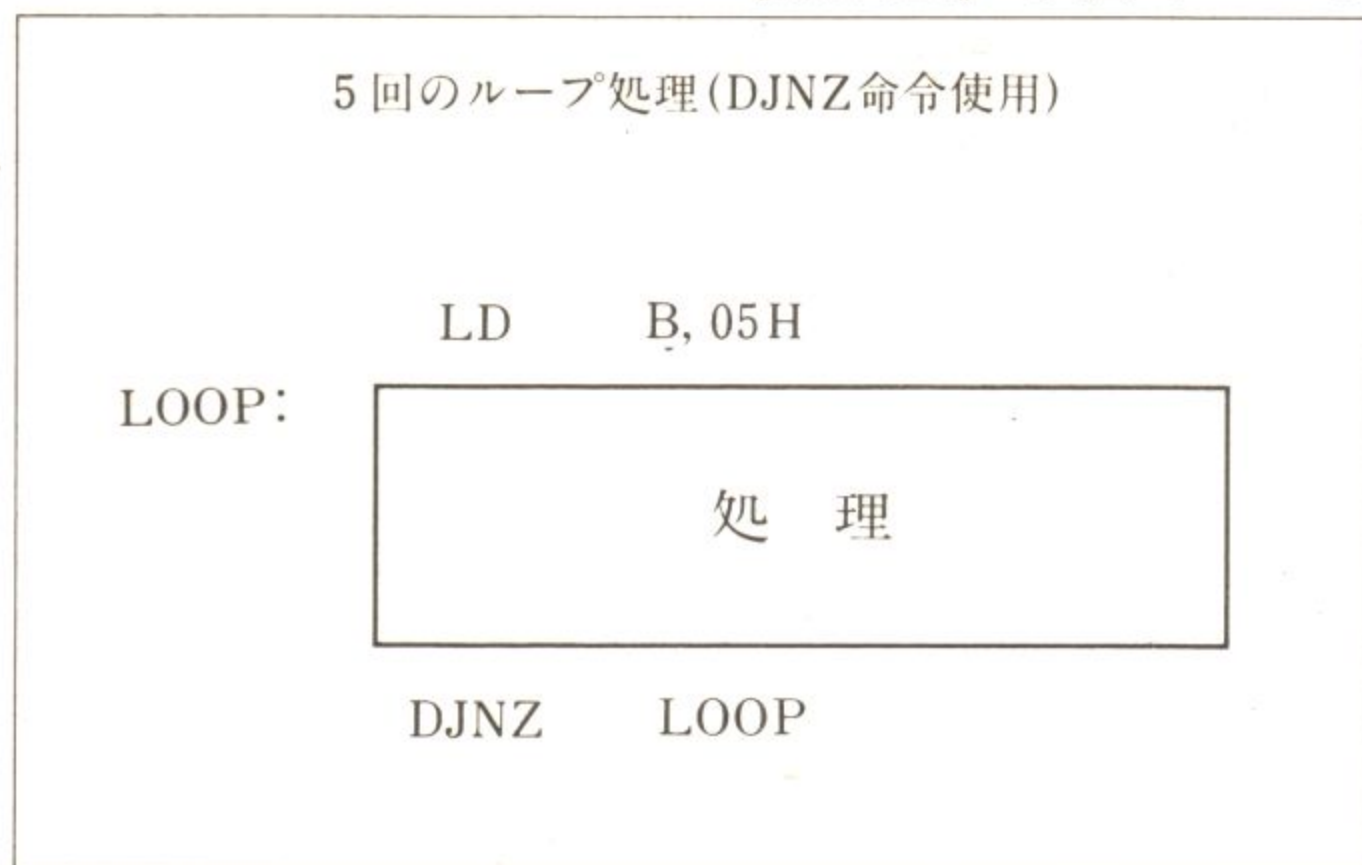
RET（サブルーチン・リターン）

を説明していないのですが興味のある方は、ながめてみてください。

《第53図》計算型ジャンプ

| HLレジスタの内容 | ジャンプ先アドレス | 実際のプログラム例 | RET命令を使用したプログラム例 |
|-----------|-----------|--------------|------------------|
| 0000H | E000H | ADD HL, HL | ADD HL, HL |
| 0001H | E010H | ADD HL, HL | ADD HL, HL |
| 0002H | E020H | ADD HL, HL | ADD HL, HL |
| 0003H | E030H | ADD HL, HL | ADD HL, HL |
| 0004H | E040H | LD DE, E000H | LD DE, E000H |
| 0005H | E050H | ADD HL, DE | ADD HL, DE |
| } | } | J P (HL) | PUSH HL |
| | | | RET |

《第54図》 5回のループ処理(DJNZ命令について)



◎Bレジスタから1を減じゼロ・フラグ(Z)が
0の時に相対ジャンプを行う命令

ループ処理を効率良く行うために、

DEC B

と、

JR NZ, ADDRESS

を組み合わせた、非常にめずらしい命令で、このタイプの命令が用意されているのは、8ビットではZ80ぐらいでしょう。

ニーモニックは、

DJNZ ADDRESS

となり、これは、

Decrement and Jump on Not Zero

の略でしょう。

なお、この命令のディクリメントの対象となるのはBレジスタのみで、又、ジャンプ先のアドレスは通常の相対ジャンプ命令と同じ様に、2バイト目で指定を行います。1バイト目は、10Hです。

参考のため、第54図に5回のループ処理を行う例をあげておきます。

まとめ

本章では、相対ジャンプ命令を中心に説明しましたが実際に初心者の方が、ジャンプ命令を使用する場合には、前章にて紹介した普通の絶対ジャンプ命令で十分だと思います。かなり簡潔に説明したため、多少わかりにくかったかも知れませんが、本書を読み終えた時点でもう一度読み直してくだされば良いと思います。

BASIC ケンちゃん

ホップ

お1ブロック



ステップ

お2ブロック

お3ブロック



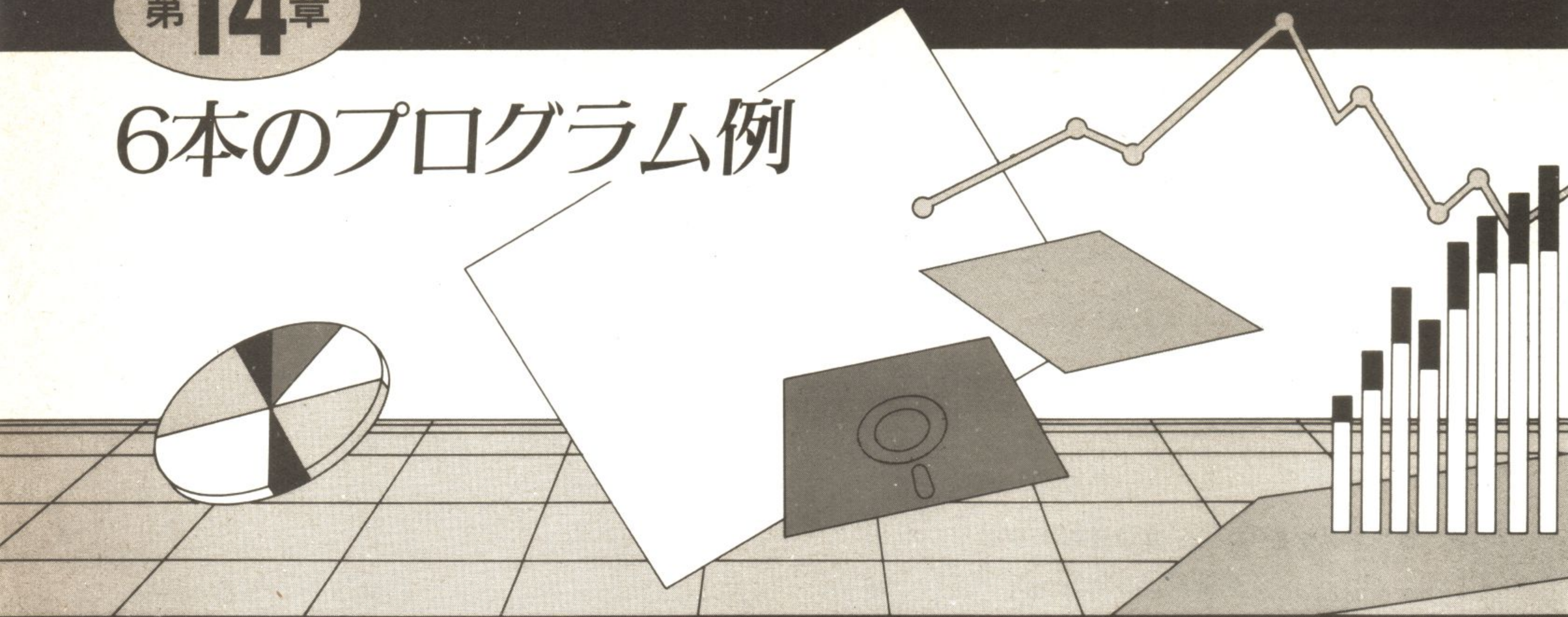
マシン語
バンザイ

ジャンプ

お4ブロック

お5ブロック

6本のプログラム例



前章までにジャンプ命令を説明し終えて一応の区切りがついた所で、本章では、

プログラム例特集

という事で、第2ブロックのまとめを行い、各命令紹介のつづきは第3ブロックで行います。

プログラム例は、

EXAMPLE 0～EXAMPLE 5

の6本を紹介しますが、EXAMPLE 0から少しずつ発展させて、最も複雑なものが、EXAMPLE 5となっています。最も複雑今まで説明し終えた数少ない命令のみで組んでありますから、容易に理解できるプログラムばかりでしょう。

また、各プログラムには簡単にですが説明を加えておきましたので、プログラム・リストと共に御一読ください。きっとアセンブラ・プログラミングの容易さに気付き、御自分のプログラムを組んでみたくなる事でしょう。

命令各々の意味がわかって、アルゴリズムが組み立てられなくては意味がありません。

早速、6本のプログラムを順に見て行きましょう。

EXAMPLE 0

簡単なもので、今までにも何度か登場して来ましたし、非常に短いプログラムですが、ここでは、アセンブラで打ち出したアセンブル・リストの見た方を練習する意味で、もう一度説明しておきま

しょう。

リスト中で、セミコロン「;」が付いている後は、全て、コメントで、実際のプログラムとは一切関係ありません。BASICのREM文か「'」だと思えば良いわけです。中にはリストを見易くするために、セミコロン「;」だけの行もあります。

また、各々のプログラムの最初に必ず置いてある「ORG」によって、プログラムを、どのアドレスからにアセンブルするかを決定します。

例えば、

```
ORG 0A000H
```

と書いて有れば、オブジェクト・プログラムは、A000H番地から、生成されるわけです。

アセンブラでは、ジャンプ先のアドレスにラベルを付けられるかわりに、16進数の頭が英文字で始まる場合には、全て、「0」を付けて、ラベルでない事を宣言しなければなりません。だから先の「A000H」も「0A000H」と、余分な「0」を付けてあるのです。

また、アポストロフィー「'」で、キャラクタを囲えば、アセンブルする際に、自動的にそのキャラクタのキャラクタ・コードに変換されます。例えば、

```
LD A, '●'
```

と、

```
LD A, 0ECH
```

とは、同じ事で、両者とも、Aレジスタに、EC

《EXAMPLE 0》

```

; *****
; *** EXAMPLE 0 ***
; *****
;
;          ORG    0A000H
;
A000 3EEC          LD    A, '●'          ; 100 A=ASC("●")
A002 2100F3        LD    HL, 0F300H      ; 110 HL=&HF300
A005 77            LD    (HL), A         ; 120 POKE HL, A
A006 C3665C        JP    5C66H          ; 130 END

```

《EXAMPLE 1》

```

; *****
; *** EXAMPLE 1 ***
; *****
;
;          ORG    0A000H
;
A000 3EEC          LD    A, '●'          ; 100 A=ASC("●")
A002 2100F3        LD    HL, 0F300H      ; 110 HL=&HF300
A005 0650          LD    B, 80           ; 120 B=80
A007 77            L1: LD    (HL), A      ; 130 POKE HL, A
A008 23            INC    HL             ; 140 HL=HL+1
A009 05            DEC    B              ; 150 B=B-1:Z=(B=0)
A00A C207A0        JP    NZ, L1          ; 160 IF NOT Z THEN 130
A00D C3665C        JP    5C66H          ; 170 END

```

Hを入れる事を意味します。アセンブラによっては、アポストロフィー「'」のかわりに、ダブル・クォーテーション・マーク「"」を用いて、

```
LD    A, "●"
```

と、表現するものもあります。

他に、アセンブラ特有の表現で利用頻度の高いものとしては、「EQU」が有りますが、ここでは使用をさけました。

さらに、プログラム全てにコメントとして、同じ事をN-BASICで行う際のプログラム・リストを入れておきました。最近この方法が流行しているようですが、マシン語のレジスタとBASICの変数の違いをしっかりと理解した上で比較するのであれば、たいへん解り易い方法と言えます。実際にコメント中のプログラム・リストを、N-BASICから打ち込めば、全て、マシン語と同じ動作を行います。もちろん実行速度は、マシン語とは問題に無らないぐらい違いますから、BASICのプログラムでゆっくり実行させ、マシン語の方が正常に動いているかを確認する事もできます。

以上、アセンブル・リストについて簡単に説明しましたが、とにかく良いアセンブラを使ってみる事です。もっとも、マシン語より使い方の解りにくいアセンブラも多いですから、アセンブラ購入時の選択には十分に気をつけてください。

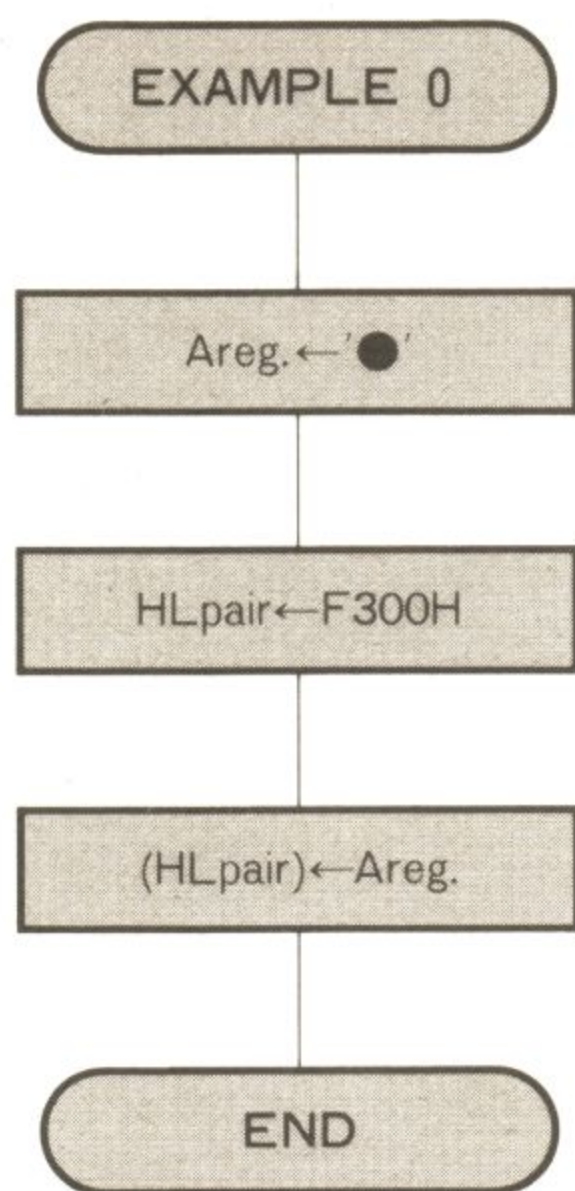
さて、EXAMPLE 0ですが、別に説明する程のプログラムでは無いでしょう。結果としては、F300H番地にECHを入れるため、CRT画面の左上端に「●」が表示されるはずですが。

EXAMPLE 1

画面の最上段1列に、80個の「●」を表示するプログラムで、ループ処理の最も簡単な例ではないでしょうか？ このプログラムでも、ビデオ・ラム(V-RAM)に直接、キャラクタ・コードを入れていきますから、40字モードの時には、1個おきに40個の「●」が表示されることになります。

アドレス指定にはHLレジスタ対を、ループ回数のカウンタにはBレジスタを使っているため、

《第55図》EXAMPLE 0 フローチャート



Hレジスタ対の値は、インクリメント命令によって、

F 3 0 0 H → F 3 0 1 H → … → F 3 4 F H
と増加し、逆にBレジスタの値はデクリメント命令で、

8 0 → 7 9 → 7 8 → … → 1 → 0

と、減少して行き、Bレジスタの値が0になった時点でループ中からぬけ出て、PC-8001のマシン語モニタへジャンプするようになっています。

この、Bレジスタのデクリメントとジャンプ命令を合わせて「DJNZ」1命令で代用する事も出来ませんが、ここでは解り易いように全て絶対ジャンプ命令を使いました。

EXAMPLE 2

このプログラムも、EXAMPLE 1と同じ様にループ処理の使用例ですが、横ではなく、画面の最左列に縦に25個の「●」を表示するためのものです。

プログラムの構成自体は、前のものとほとんど変わっていないのですが、ループ回数のカウントにはCレジスタを用いました。

また、横に並べる場合にはメモリ指定に使っているHLレジスタ対の値を、インクリメント命令によって1ずつ増して行けば良かったのですが、縦の場合には画面80文字分にアトリビュート・エリア40バイト分を加えた120バイト分HLレジスタに加えて行かなければなりません。

そこで、DEレジスタ対にあらかじめ120を入れておいて、1ループごとに、16ビット加算命令を用いて、HLレジスタ対にDEレジスタ対の値を加えているのが、

ADD HL, DE

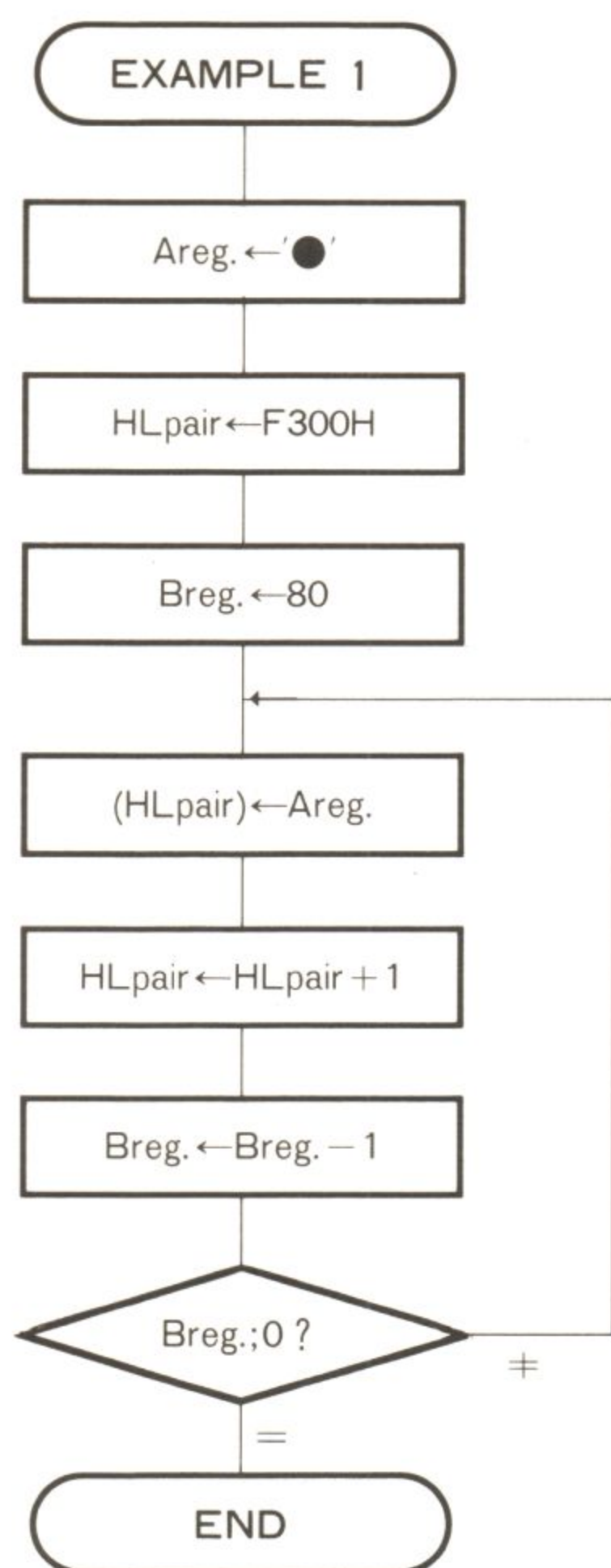
です。ループ処理からは、Cレジスタが0になった25回目にぬけ出ています。

《EXAMPLE 2》

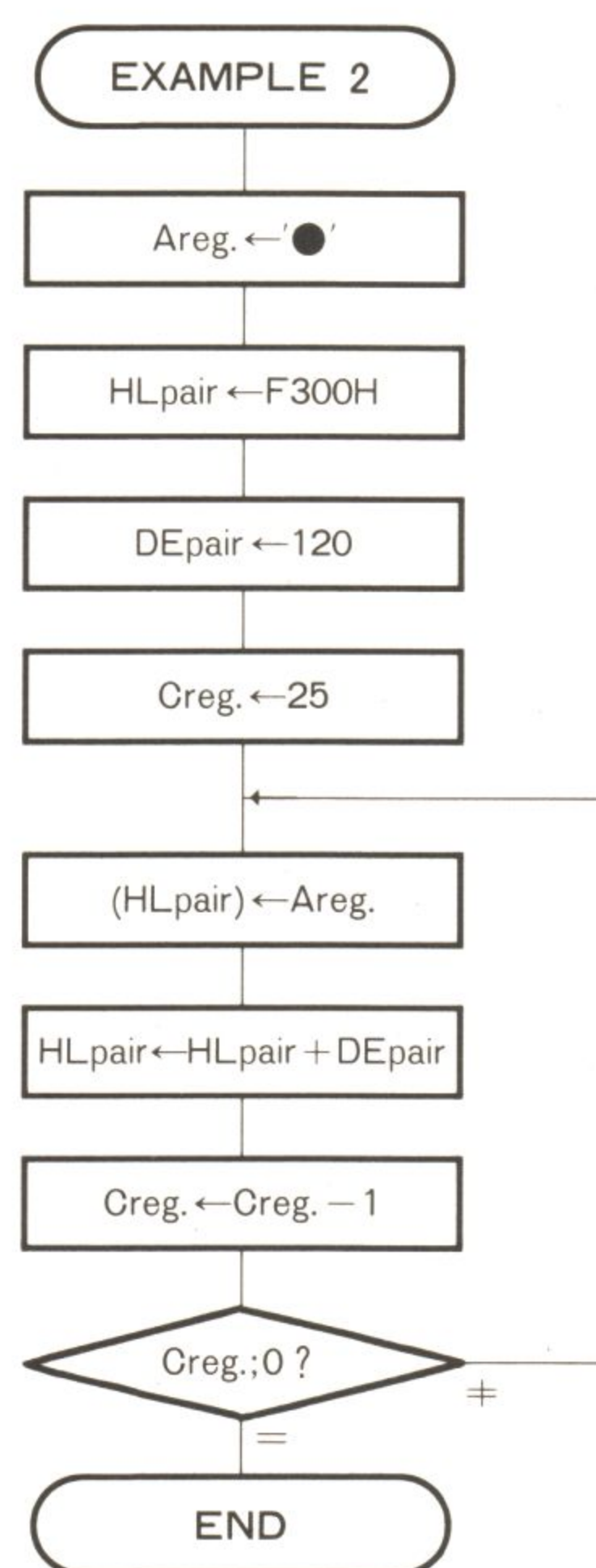
```

; *****
; *** EXAMPLE 2 ***
; *****
;
; ORG 0A000H
;
A000 3EEC          LD  A, '●'          ; 100 A=ASC ("●")
A002 2100F3        LD  HL, &HF300H      ; 110 HL=&HF300
A005 117800        LD  DE, 120          ; 120 DE=120
A008 0E19          LD  C, 25            ; 130 C=25
A00A 77            L2: LD  (HL), A       ; 140 POKE HL, A
A00B 19            ADD  HL, DE           ; 150 HL=HL+DE
A00C 0D            DEC  C               ; 160 C=C-1: Z= (C=0)
A00D C20AA0        JP  NZ, L2           ; 170 IF NOT Z THEN 140
A010 C3665C        JP  5C66H           ; 180 END
    
```


《第56図》EXAMPLE 1 フローチャート



《第57図》EXAMPLE 2 フローチャート

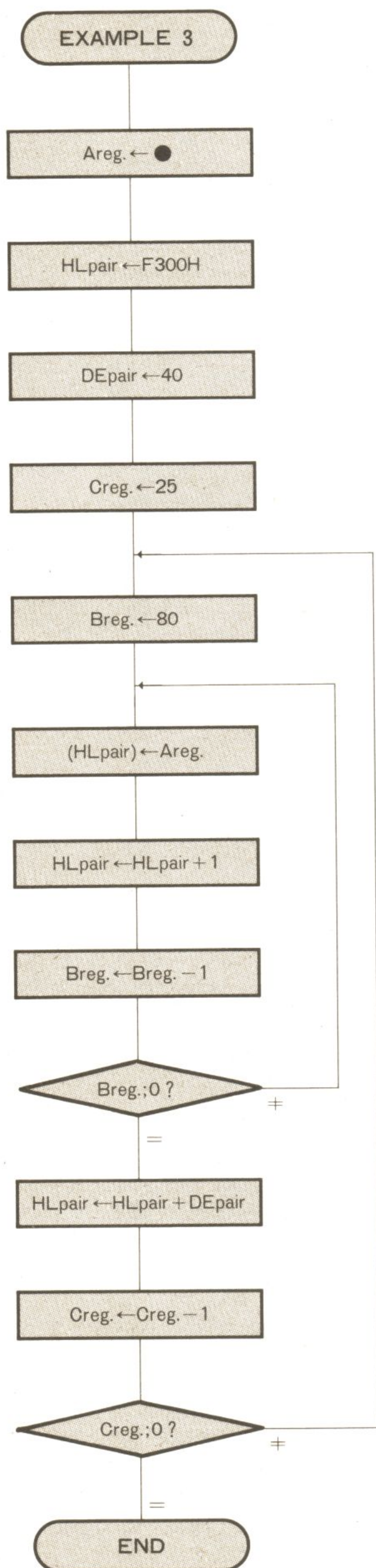


《EXAMPLE 3》

```

; *****
; *** EXAMPLE 3 ***
; *****
;
;          ORG    0A000H
;
A000 3EEC          LD    A, '●'          ; 100 A=ASC ("●")
A002 2100F3        LD    HL, 0F300H      ; 110 HL=&HF300
A005 112800        LD    DE, 40          ; 120 DE=40
A008 0E19          LD    C, 25           ; 130 C=25
A00A 0650          L3:  LD    B, 80       ; 140 B=80
A00C 77            L4:  LD    (HL), A     ; 150 POKE HL, A
A00D 23            INC    HL              ; 160 HL=HL+1
A00E 05            DEC    B              ; 170 B=B-1: Z= (B=0)
A00F C20CA0        JP     NZ, L4          ; 180 IF NOT Z THEN 150
A012 19            ADD    HL, DE          ; 190 HL=HL+DE
A013 0D            DEC    C              ; 200 C=C-1: Z= (C=0)
A014 C20AA0        JP     NZ, L3          ; 210 IF NOT Z THEN 140
A017 C3665C        JP     5C66H           ; 220 END
  
```


《第58図》EXAMPLE 3 フローチャート



EXAMPLE 3

EXAMPLE 1と2ではどちらも1回ずつのループ処理を行っていましたが、両者を組み合わせたものが、EXAMPLE 3となっています。

フローチャートを見ていただいてもわかるようにループが2重になっていますが、これを、ループのネスティングと言い、場合によっては3重4重以上のネスティングを行う事もあります。

プログラムを実行すると、一瞬でCRT画面中が、「●」で埋まりますが、実際には、最上段の左端から右端へ80個の「●」を入れて行き、それを最下段までの25行分くり返す事によって画面中を埋めています。コメントとして入っている、N-BASICのプログラムを実行すれば、画面が埋められて行く状態がゆっくりとシミュレートされて良くわかると思います。

1行分80個の表示を終えた時点で、メモリ指定用のHLレジスタ対に40を加えています。これは、各行40バイトのアトリビュート・エリアをスキップするため、PC-8001独特の手法です。アトリビュート方式はこの様な場合には手間がかかりますね。

EXAMPLE 4

「●」ばかりを表示させていても面白く無いので、今度は表示するキャラクタを変化させてみました。EXAMPLE 3までのプログラムでも全て、Aレジスタに「●」のキャラクタ・コードであるECHを入れておき、HLレジスタ対の指すアドレスにAレジスタの値を移すという、2段がまえで、ビデオ・ラム(V-RAM)にキャラクタ・コードを送り込んでいますから、Aレジスタの値を変化させる事によって、簡単に、表示するキャラクタを変える事ができるのです。

Aレジスタの値は最小のループごとに、インクリメント命令で1ずつ増して行きます。ここで注意しなければならないのが、レジスタと変数の違いで、レジスタの場合にはFFH(255)を越

《EXAMPLE 4》

```

; *****
; *** EXAMPLE 4 ***
; *****
;
;          ORG  0A000H
;
A000 3E00          LD  A,0          ; 100 A=0
A002 2100F3        LD  HL,0F300H    ; 110 HL=&HF300
A005 112800        LD  DE,40        ; 120 DE=40
A008 0E19          LD  C,25         ; 130 C=25
A00A 0650          LD  B,80         ; 140 B=80
A00C 77            L5: LD  (HL),A    ; 150 POKE HL,A
A00D 3C            L6: INC  A        ; 160 A=(A+1) AND &HFF
A00E 23            INC  HL          ; 200 HL=HL+1
A00F 05            DEC  B           ; 210 B=B-1:Z=(B=0)
A010 C20CA0        JP   NZ,L6       ; 220 IF NOT Z THEN 150
A013 19            ADD  HL,DE        ; 230 HL=HL+DE
A014 0D            DEC  C           ; 240 C=C-1:Z=(C=0)
A015 C20AA0        JP   NZ,L5       ; 250 IF NOT Z THEN 140
A018 C3665C        JP   5C66H       ; 260 END

```

えると0にクリアされてしまいます。ところが、BASICでは256以上の数値を扱えますから、255を越えた時のために、常にFFHと論理積ANDを取って、下位1バイト以外は無効にしなければなりません。さもないと、変数Aの値が255を越えた直後の、

POKE HL, A

の部分で、

Illegal function call

のエラーが発生してしまいます。これは、メモリには1バイトを越える数値を入れる事が出来ないからです。

以上の様に、バイト・データを扱っている場合にはBASICよりマシン語の方が簡単な事も有り得るのです。

EXAMPLE 5

このプログラムは、実際に実行させた後、ぜひ御自分で解析してみてください。

たった30バイト程度のプログラムで、これだけすばらしい動きを持ったデモンストレーションが出来るのですから、やはりマシン語は偉大です。

実際には、EXAMPLE 4をわずかに改良

しただけですが、全く異なった実行結果となっています。

スタックへのレジスタ退避なども行っていますが、フローチャートを見ながらプログラム・リストを追って行けば理解できると思います。

なお、プログラムは無限ループに入ってしまうので止めたい場合には、リセットをかけてください。

フローチャートについて

フローチャートは、各プログラム相互の関係と流れがわかり易い事に重点をおいてあります。

フローチャート中で、矢印「←」は代入する事を表わし、「reg.」は1バイトのレジスタに、「pair」はレジスタ対に付けました。

また、レジスタ対をカッコではさんだものはニーモニックと同じく、そのレジスタ対が指すメモリのアドレスを表わします。

少しずつ複雑になって行く様子を楽しんでください。

《EXAMPLE 5》

```

; *****
; *** EXAMPLE 5 ***
; *****
;
;          ORG 0A000H
;
A000 3E00          LD  A,0          ; 100 A=0
A002 F5            L7:  PUSH AF      ; 110 STACK=A
A003 2100F3        LD  HL,0F300H    ; 120 HL=&HF300
A006 112800        LD  DE,40       ; 130 DE=40
A009 0E19          LD  C,25        ; 140 C=25
A00B 0650          L8:  LD  B,80     ; 150 B=80
A00D 77            L9:  LD  (HL),A  ; 155 POKE HL,A
A00E 3C            INC  A           ; 160 A=(A+1) AND &HFF
A00F 23            INC  HL          ; 170 HL=HL+1
A010 05            DEC  B           ; 180 B=B-1:Z=(B=0)
A011 C20DA0        JP   NZ,L9      ; 190 IF NOT Z THEN 155
A014 19            ADD  HL,DE       ; 195 HL=HL+DE
A015 0D            DEC  C           ; 200 C=C-1:Z=(C=0)
A016 C20BA0        JP   NZ,L8      ; 210 IF NOT Z THEN 150
A019 F1            POP  AF          ; 220 A=STACK
A01A 3C            INC  A           ; 230 A=(A+1) AND &HFF
A01B C302A0        JP   L7         ; 240 GOTO 110

```

まとめ

第14章では、6本のプログラム例を紹介しましたが、いかがだったでしょうか？

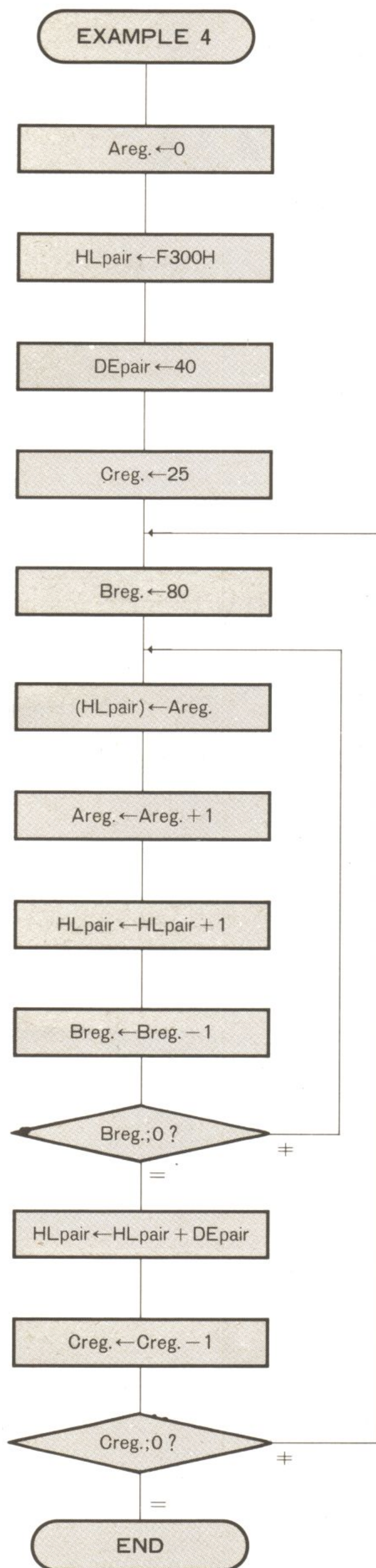
自分でプログラムを組んでみる事が、マイクロ・コンピュータを理解するために一番重要なのは言うまでもありませんが、そのためにも他人のつくったプログラムを解析して、プログラミングの定石を学びとる事が必要かと思います。

プログラム例には、それほど詳しい説明も付けず、皆さんの解析へのヒントだけを簡単に加えておいたつもりですから、マシン語でのプログラミングの定石をつかんで、御自分のプログラミングに役立ててください。

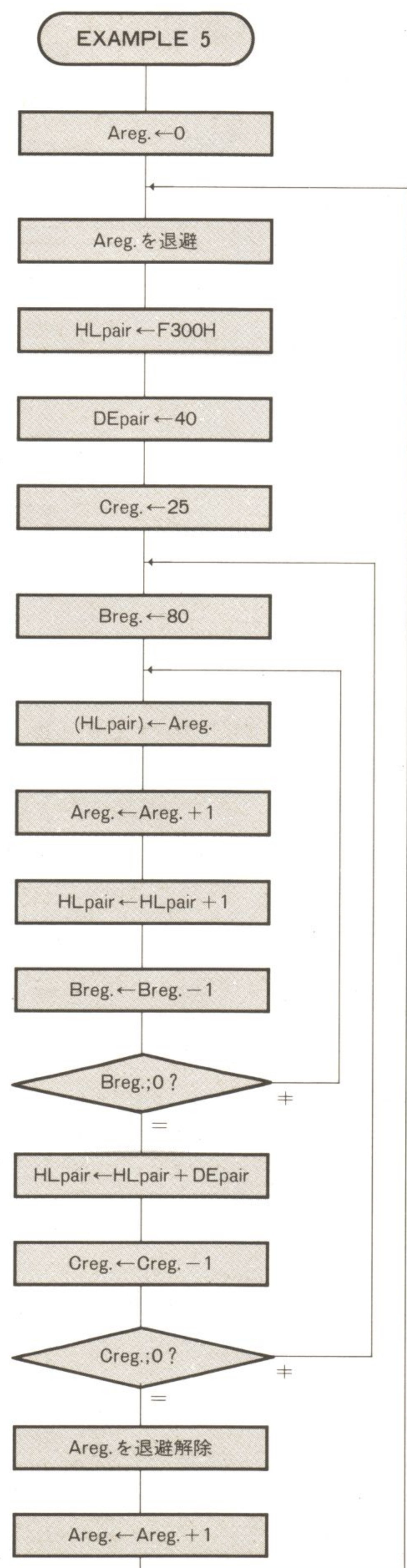
次の第15章からは、今までどおりに、各命令の紹介にもどり、サブルーチン・コール命令から続けて行きます。



《第59図》EXAMPLE 4 フローチャート



《第60図》EXAMPLE 5 フローチャート



CPU マイクロプロセッサの流れ

8ビット機の時代

8080と6800

コンピュータの端末用に開発された8008の発表後、潜在需要の大きさを感じはじめた各半導体のメーカーは、より汎用化し能力を強化したCPUの開発に着手し、1974年にはインテル社が8080(i8080)を、モトローラ社が6800(M6800)を、フェアチャイルド社がF-8を発表したことにより、マイクロプロセッサというものが業界で注目を受けはじめました。

インテル社の8080は、8008の改良版であったにもかかわらず、それまでの専用機から脱却し、急速に8ビット・マイクロプロセッサの標準機としての立場を築いてしまいました。これは、8080がいわゆる第2世代のマイクロプロセッサとしては最初に発表され売出されたこと、市場がインテル社の主導型で進んできたことなどにも起因しているにちがいありません。

モトローラの6800は、ハードウェアの構成やインストラクション・セットが非常に単純明快で使い易い点を誰もが認めるところでしたが、8080用のオペレーティング・システムとしてデジタルリサーチ社よりCP/Mが発表され普及するにつれて、8ビット・マイクロプロセッサのシェアにおける6800の大敗は確実なものとなってしまいました。

Z80と8085

ところで、8080は日本人である嶋正利氏によって設計さ

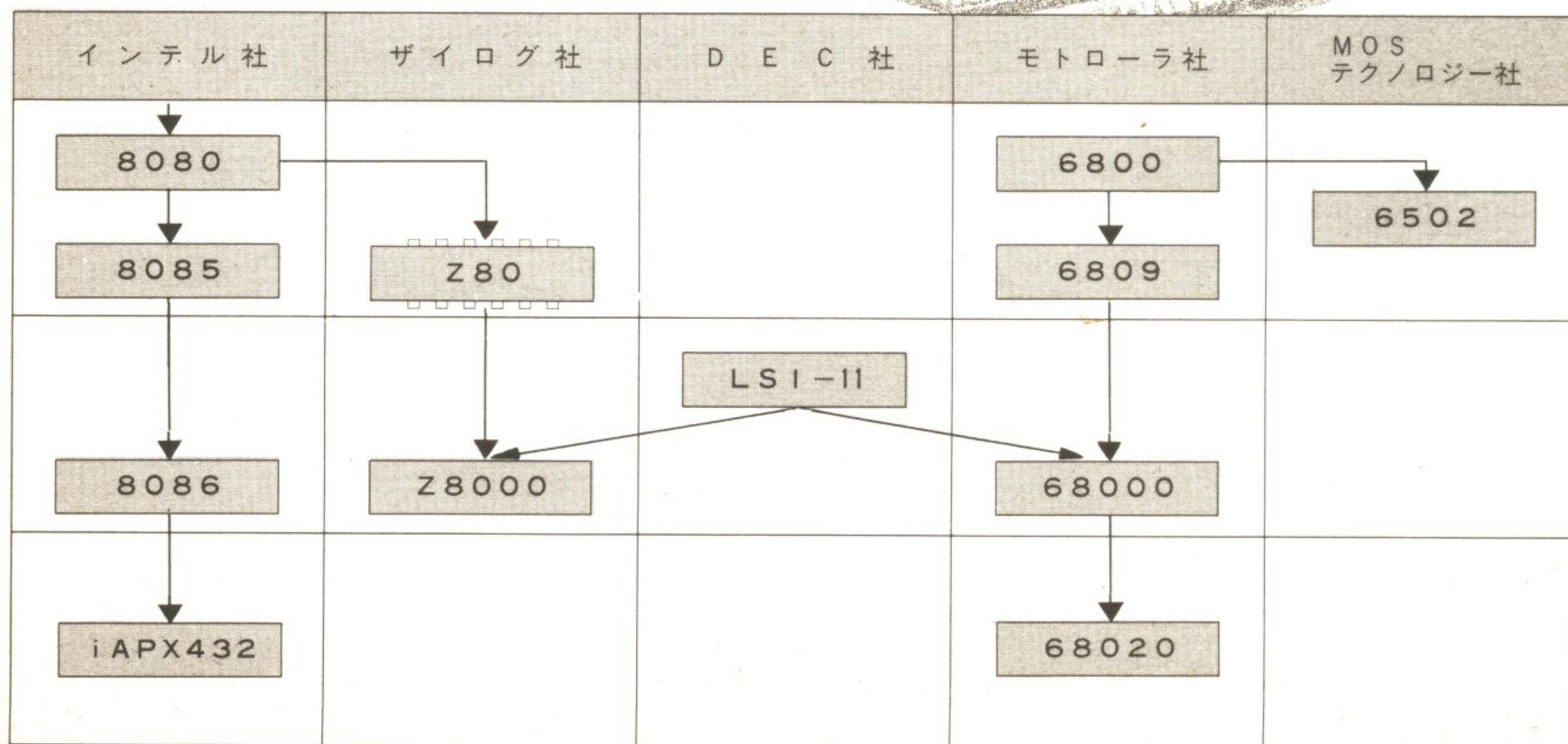
れたのですが、同氏がザイログ社に移って設計し、大手石油会社で我国ではESSOの商標で有名なエクソン社の資金力にものをいわせて開発されたものがZ80です。Z80は8080と完全なコンパチビリティ(上位互換性)を持っていたためあって圧倒的なシェアを獲得し、日本でも主要なパーソナルコンピュータのほとんどがZ80を搭載していることは皆さんも良く御存じだと思います。

インテル社でも、5ボルト単一電源でしかも8080のために用いてきた周辺ICが不要の8085を発表しZ80に対抗しました。

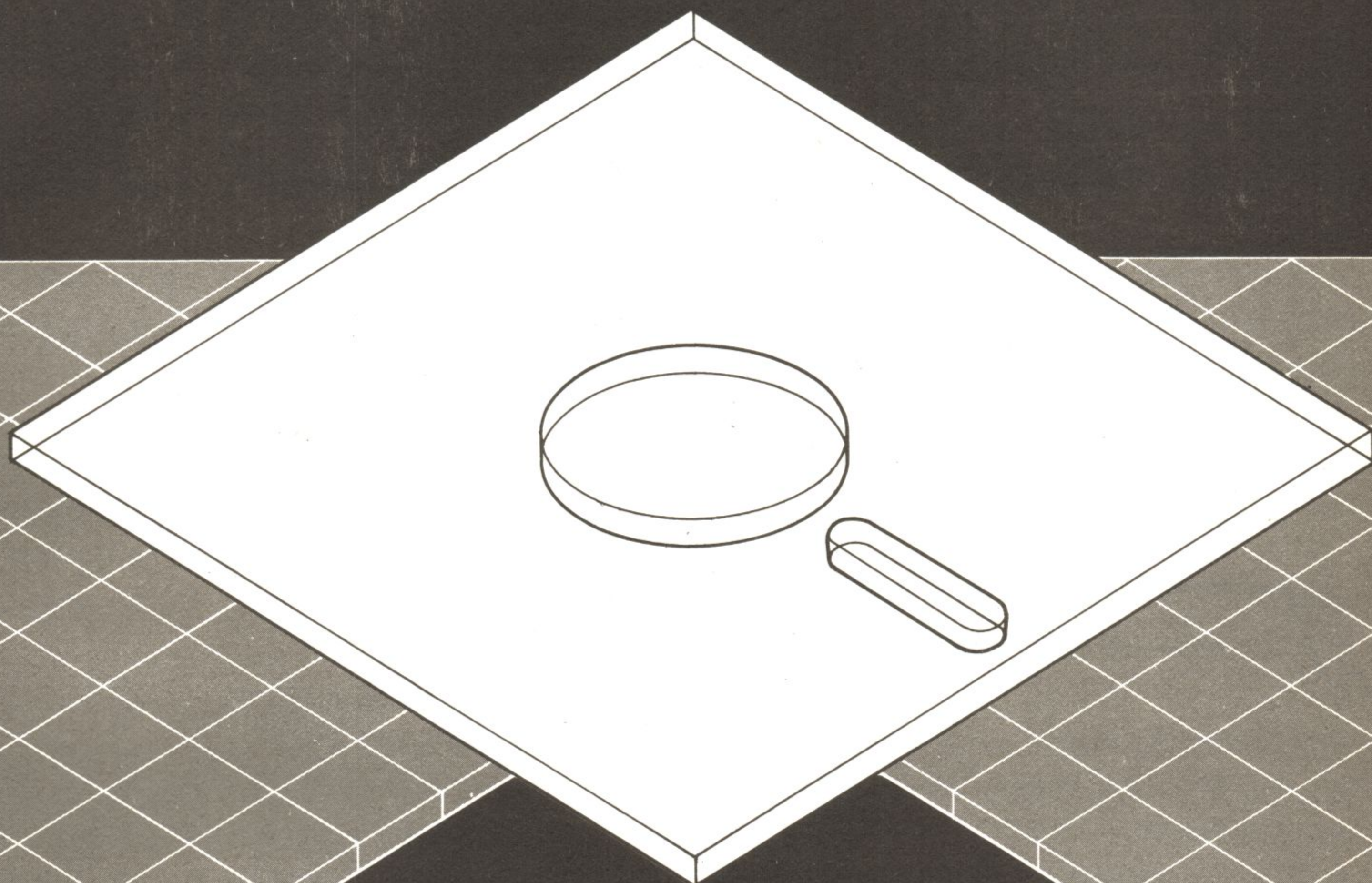
6502と6809

6800系では、MOSテクノロジー社が6800の流れを継いだ6502を発表しましたが、このCPUは、コモドール社のPETシリーズやCBMシリーズ、アップル社のAPPLE II等、多くの米国製パーソナル・コンピュータに採用されました。

さらにモトローラ社自身も、6800の上位機である6809(MC6809)を発表しました。6809はシェアこそ及びませんでしたが、8ビットのCPUとして考えられ得るほとんどすべての機能が盛り込まれ、究極の8ビット・マイクロプロセッサとさえ呼ばれています。また、6809の基では、OS-9という非常にすぐれたオペレーティング・システムを走らせることができます。

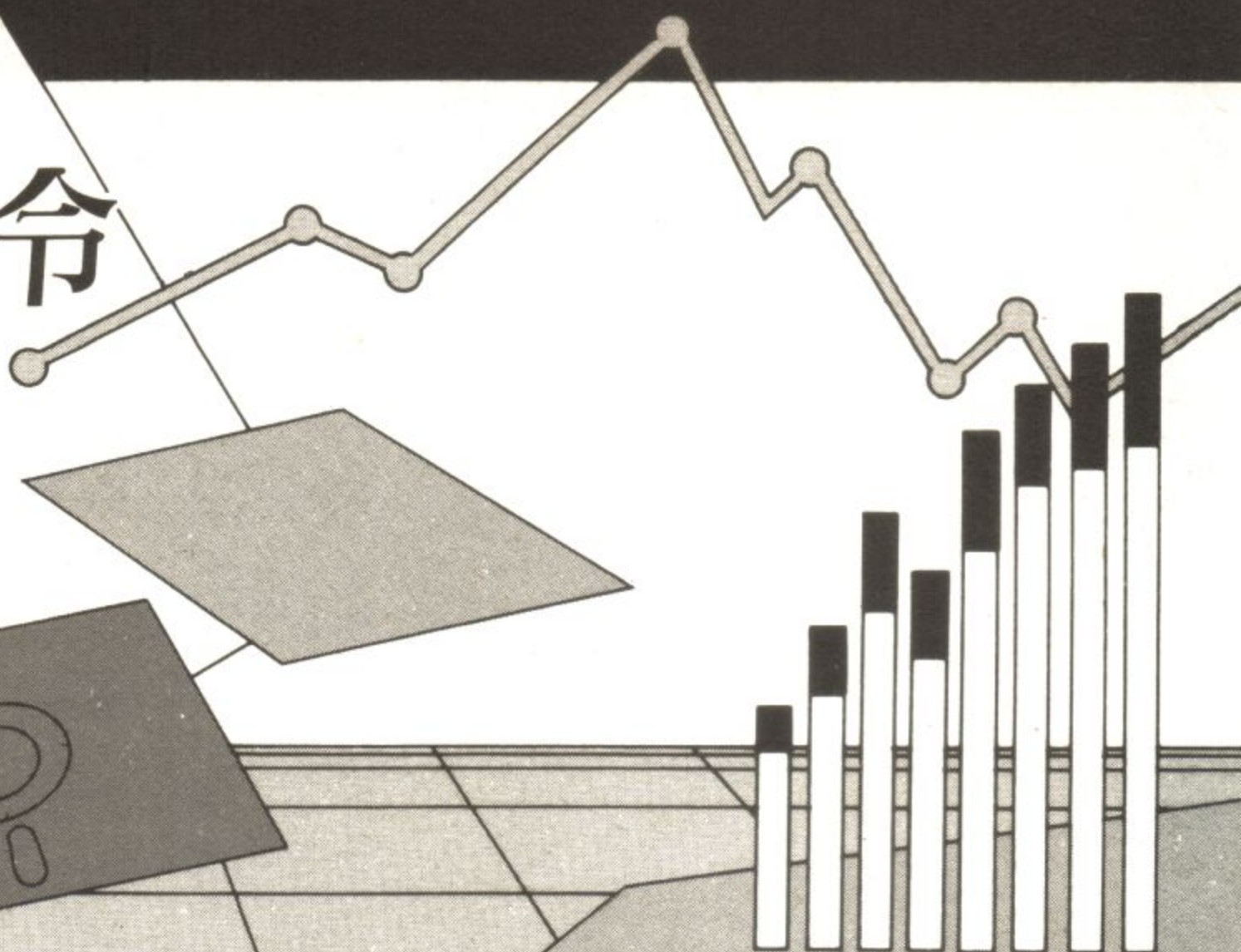
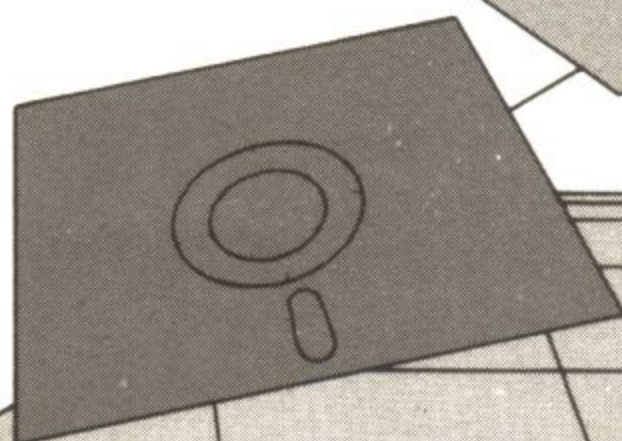
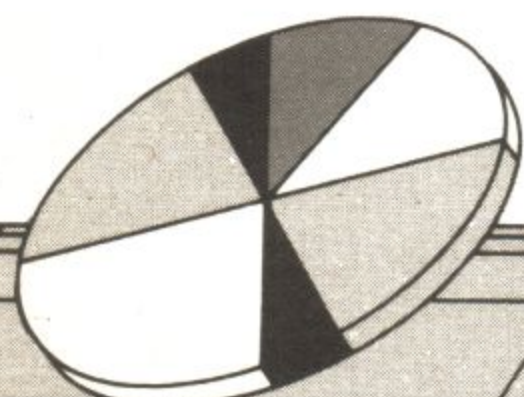


第3ブロック



Z80の
インストラクション・セットII

サブルーチン・コール命令 ／リターン命令



本章では、マシン語でサブルーチンをつくる方法を説明します。説明の内容も第2ブロックまでの命令中心から、実例中心へと少しずつ変えて行くつもりですからマシン・コード・プログラミングの定石を盗み取るつもりで、しっかり読んでください。

サブルーチンの使用例

本章で説明する、サブルーチン・コール命令とリターン命令は、共にサブルーチンを使用する時に用いる命令です。

サブルーチンについては、BASIC等を通じて御存知の方も多いと思いますが、簡単な例をあげておきましょう。

たとえば、2バイトの16進数を1000倍するプログラムを考えてみましょう。いくつかの方法が考えられますが、私が考えたのは、10倍する事を3回くり返す事によって、1000倍を実現する方法で、

$$1000 = 10^3$$

である事を利用した方法です。

1000倍するための被乗数を与える事が必要ですが、これはE000H～E001H番地に渡って入れておく事にし、積はE002～E003H番地に収納しますから、演算結果を調べるにはマシン語モニタのDコマンドなどを使用します。

また、演算した結果が16ビットを超えた場合に無視する様にしましょう。つまり、あまり大きな数を1000倍することはできません。

以上の仕様によって、プログラミングを行うのですが、さしあたって問題になるのが、2バイトの整数を10倍する方法でしょう。実際に乗算を行うのは乗算命令の用意されていないCPUには負担ですから加算命令だけで間に合うように考えてみましょう。

HLレジスタ対の値を10倍する

$$HL \leftarrow HL * 10$$

を、次のように変形してみました。

$$HL \leftarrow ((HL * 2) * 2 + HL) * 2$$

これなら、加算命令のみで何とか間に合いそうです。早速10倍するルーチンを組んでみましょう。

```
LD    D, H
LD    E, L
ADD   HL, HL
ADD   HL, HL
ADD   HL, DE
ADD   HL, HL
```

HLレジスタ対の値を保存しておくために、DEレジスタ対を使ってしまいましたが、問題は無いでしょう。

上記の10倍ルーチンを3回連続して使い、E000H～E001H番地に渡る16進数データを1000倍するプログラムを組んだものが、EXAMPLE-0です。モニタのGコマンドで実行すれ

ば、演算結果がE002H～E003番地に収納されますので試してみてください。

このEXAMPLE-0を良く見ると、全く同じ部分が3箇所あります。当然の事ながら10倍ルーチンがそのまま3回使われていますね。この、10倍ルーチンを1箇所分だけ用意しておいて3箇所から呼び出して使おうと考えて組み直したプログラムが、EXAMPLE-1です。EXAMPLE-0より、だいぶすっきりしていると思いませんか？

EXAMPLE-1の中で使用されている

CALL ADDRESS

が、サブルーチン・コール命令で、

RET

は、リターン命令です。N-BASICでは、

GOSUB 行番号

RETURN

に相当するでしょうか。

そして、10倍ルーチンを、サブルーチン (Subroutine) と呼び、マシン語のプログラムでは非

常に多くのサブルーチンを使うのが普通です。

EXAMPLE-0とEXAMPLE-1の関係を簡単な図(第61図～第62図)に直して挙げておきますので、両者を比較してサブルーチンの基本的な使い方をよく理解しておいてください。なおEXAMPLE-2は、さらにループ処理を行ったものです。普通はEXAMPLE-1の段階で十分なのですが、参考にあげておきます。

コール命令<及び> リターン命令

サブルーチン・コール命令、リターン命令には、無条件コール命令、無条件リターン命令と条件コール命令、条件リターン命令の2種類があります。前者は、サブルーチンの使用例でも使用した様

《EXAMPLE-0》

| | |
|----------------------------|----------------|
| ;===== | |
| ; program : EXAMPLE-0 | |
| ;===== | |
| ; FUNCTION := X1000 | |
| ; INPUTS := (E000H-E001H) | |
| ; OUTPUTS := (E002H-E003H) | |
| ;===== | |
| ; ORG 0D000H | |
| ;===== | |
| D000 2A00E0 | LD HL,(0E000H) |
| D003 54 | LD D,H |
| D004 5D | LD E,L |
| D005 29 | ADD HL,HL |
| D006 29 | ADD HL,HL |
| D007 19 | ADD HL,DE |
| D008 29 | ADD HL,HL |
| ;===== | |
| D009 54 | LD D,H |
| D00A 5D | LD E,L |
| D00B 29 | ADD HL,HL |
| D00C 29 | ADD HL,HL |
| D00D 19 | ADD HL,DE |
| D00E 29 | ADD HL,HL |
| ;===== | |
| D00F 54 | LD D,H |
| D010 5D | LD E,L |
| D011 29 | ADD HL,HL |
| D012 29 | ADD HL,HL |
| D013 19 | ADD HL,DE |
| D014 29 | ADD HL,HL |
| ;===== | |
| D015 2202E0 | LD (0E002H),HL |
| ;===== | |
| D018 C3665C | JP 5C66H |

《EXAMPLE-1》

| | |
|----------------------------|----------------|
| ;===== | |
| ; program : EXAMPLE-1 | |
| ;===== | |
| ; FUNCTION := X1000 | |
| ; INPUTS := (E000H-E001H) | |
| ; OUTPUTS := (E002H-E003H) | |
| ;===== | |
| ; ORG 0D000H | |
| ;===== | |
| D000 2A00E0 | LD HL,(0E000H) |
| D003 CD12D0 | CALL SUB1 |
| D006 CD12D0 | CALL SUB1 |
| D009 CD12D0 | CALL SUB1 |
| ;===== | |
| D00C 2202E0 | LD (0E002H),HL |
| ;===== | |
| D00F C3665C | JP 5C66H |
| ;===== | |
| ; SUBROUTINE-1 | |
| ;===== | |
| ; FUNCTION := X10 | |
| ; INPUTS := HL | |
| ; OUTPUTS := HL | |
| ;===== | |
| D012 54 | SUB1: LD D,H |
| D013 5D | LD E,L |
| D014 29 | ADD HL,HL |
| D015 29 | ADD HL,HL |
| D016 19 | ADD HL,DE |
| D017 29 | ADD HL,HL |
| ;===== | |
| D018 C9 | RET |

に、無条件にサブルーチンを呼び、またサブルーチンからもどって来る命令で、後者は、フラグ・レジスタ (F) の各フラグの状態によって、サブルーチンのコールやサブルーチンからのリターンを行うための命令です。

ジャンプ命令を紹介した時には、フラグによる条件判断に不慣れな皆さんも多い事と思い、一つの命令を取りあげて説明して行きました。しかし、第14章の「プログラム例特集」なども通してフラグの変化にもだいたい慣れて来た事と思いますので、本章のコール命令とリターン命令は簡単

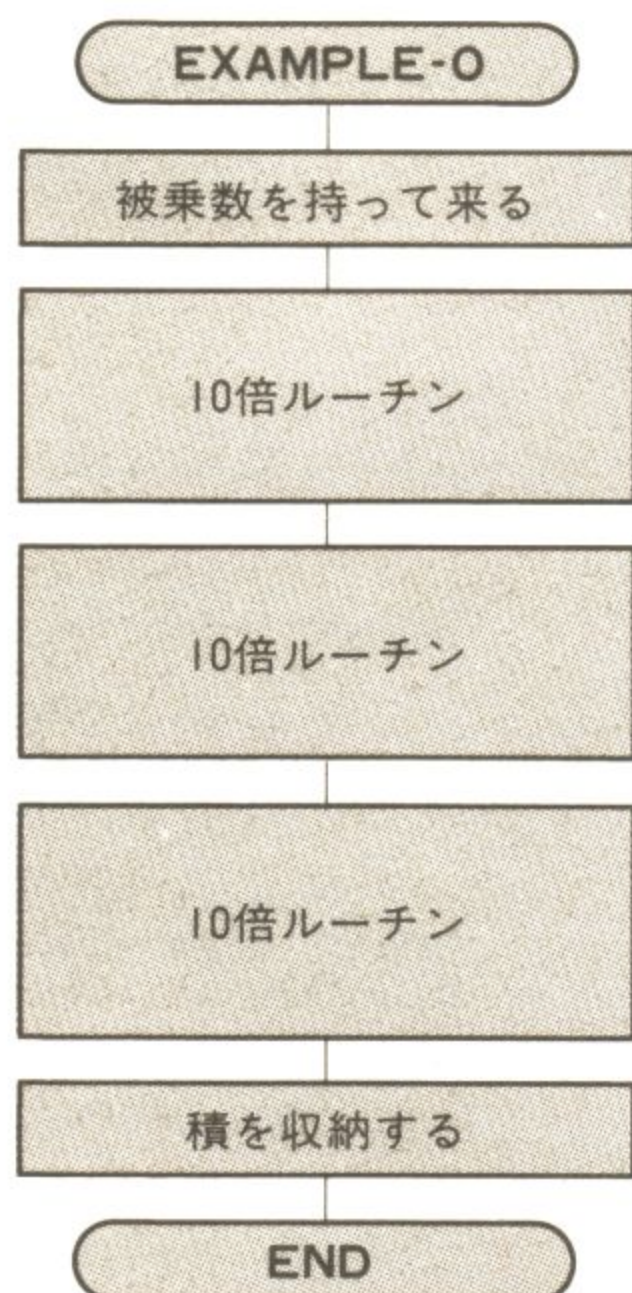
な説明と表だけを挙げておきたいと思います。あとは、より応用的な話を述べたいと思います。

《EXAMPLE-2》

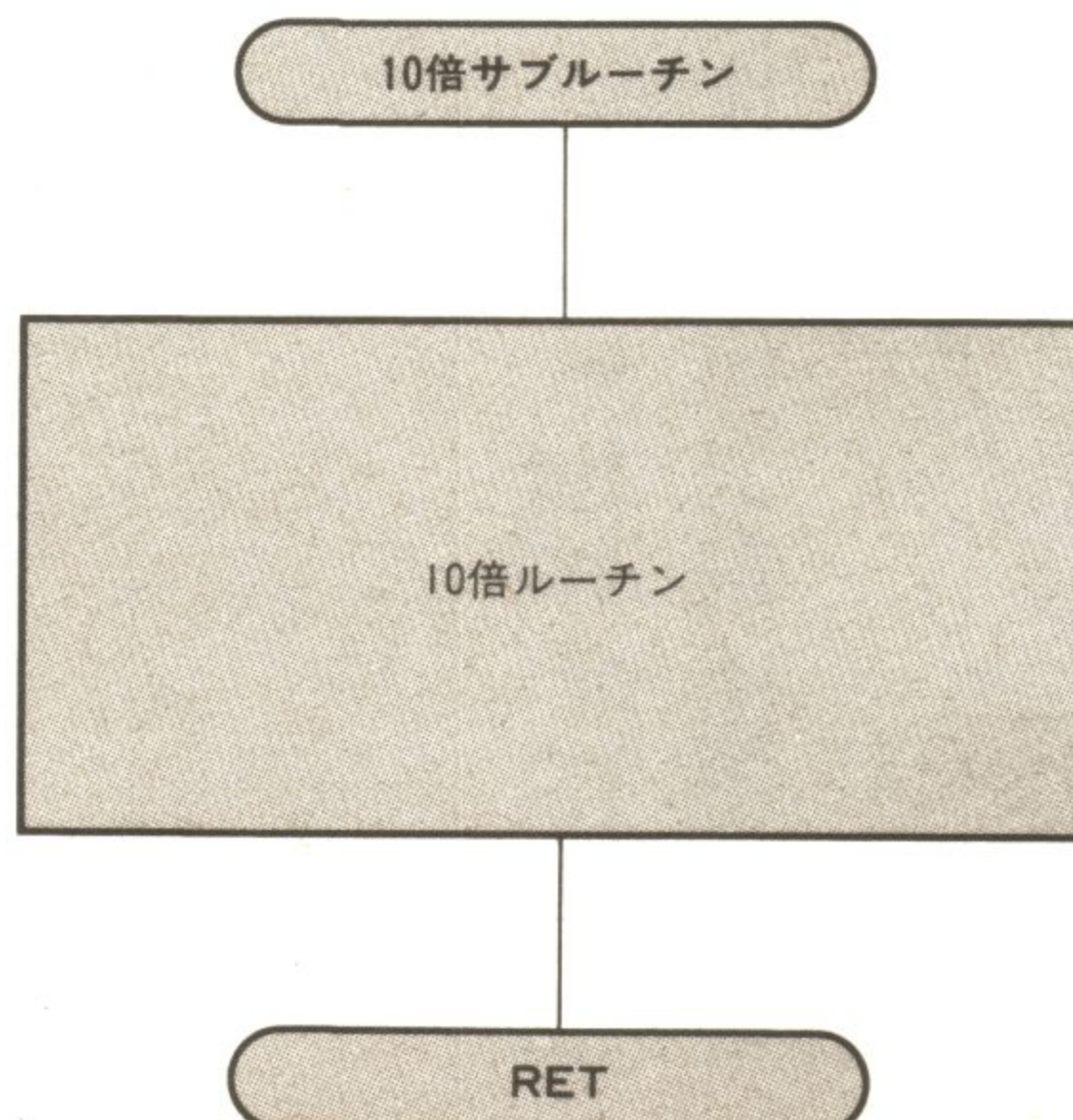
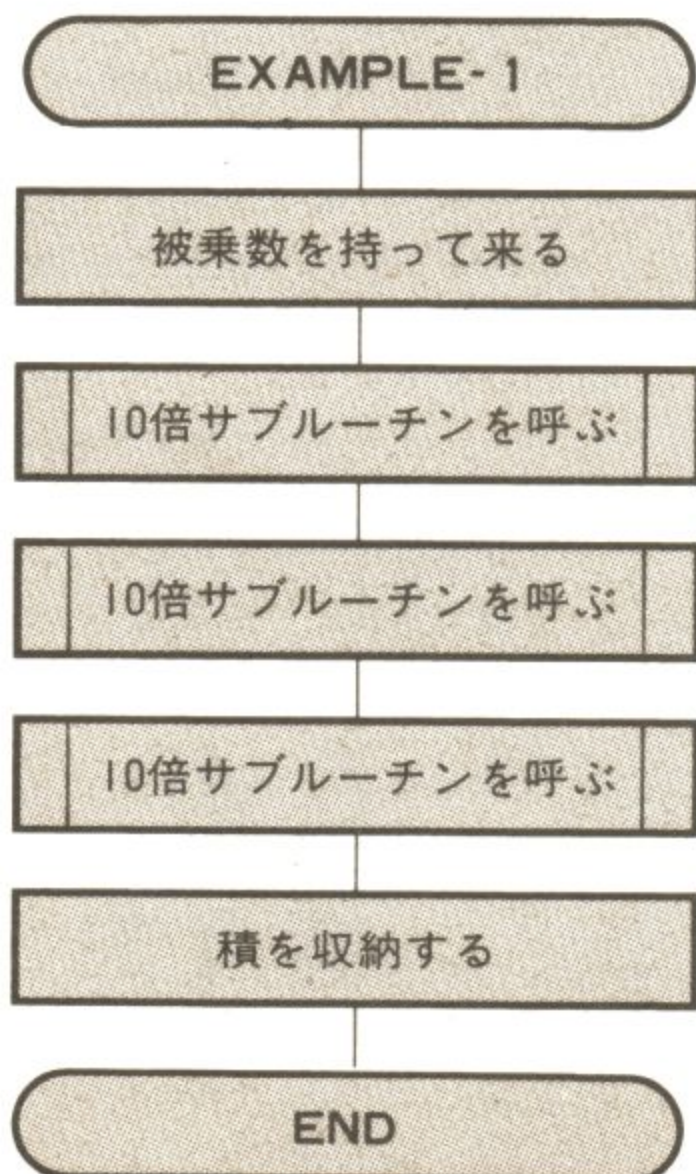
```

=====
;
;   program : EXAMPLE-2
;
=====
;
; FUNCTION := X1000
; INPUTS   := (E000H-E001H)
; OUTPUTS  := (E002H-E003H)
;
;
;   ORG 00000H
;
;
;   LD HL,(0E000H)
;
;   LD B,3
;   LOOP: CALL SUB2
;         DJNZ LOOP
;
;   LD (0E002H),HL
;
;   JP 5C66H
;
;
;-----
;
; SUBROUTINE-2
;-----
;
; FUNCTION := X10
; INPUTS   := HL
; OUTPUTS  := HL
;
;
; SUB2: LD D,H
;       LD E,L
;       ADD HL,HL
;       ADD HL,HL
;       ADD HL,DE
;       ADD HL,HL
;
;       RET
;
D000 2A00E0
D003 0603
D005 CD10D0
D008 10FB
D00A 2202E0
D00D C3665C
D010 54
D011 50
D012 29
D013 29
D014 19
D015 29
D016 C9
    
```

《第61図》 EXAMPLE-0



《第62図》 EXAMPLE-1



なお、リターン命令は全て1バイト命令、コール命令は全て3バイト命令で、2～3バイト目でジャンプ命令などと同じ様に、コールするサブルーチンのアドレスを指定します。

以前説明したジャンプ命令に、ここで登場したサブルーチン・コール命令、リターン命令などを加えた、ニーモニック⇔マシン語の対応表(第63図)をあげておきますので、どんな命令があるのかを見ておいてください。もっとも、条件コール命令、リターン命令を使う機会はそれほど多くなく、ほとんどは無条件コール命令、リターン命令のみでも十分によろしく足りる事でしょう。

CPU側から見た サブルーチン・コール

コール命令によってサブルーチンを呼び、リターン命令でもどって来る事は、容易に理解してもらえたと思います。ところで、このサブルーチン・コールにおいて、Z80は実際にはどのような働きを行っているのでしょうか？ 無条件コール命令、リターン命令を例に取って考えてみましょう。

まず、サブルーチン・コール時にCPUが行う事は、サブルーチンから帰る時のために、現在の

プログラム・カウンタ(PC)の値を退避する事です。退避場所には、レジスタ退避の時に使われたのと同じスタックが用いられます。プログラム・カウンタ(PC)の値をスタックに退避した後は当然スタック・ポインタ(SP)からも2を減じておきます。次にジャンプ命令と同じ様に、コール先のサブルーチンのアドレスがプログラム・カウンタ(PC)に送り込まれてサブルーチンにジャンプします。

リターン命令ではこれと全く逆の動作を行い、スタックに退避してあった、もどり先のアドレスをスタックから取り出して、プログラム・カウンタ(PC)に送り込む事によってサブルーチンからもどる事ができます。

以上の事から、サブルーチン・コール命令、リターン命令共に、ジャンプ命令の特殊なものと考えられます。

また、サブルーチン内ではスタックの使い方に注意しなければならない事もわかります。スタックの内容を壊した上でリターン命令を実行してしまうと、プログラム・カウンタ(PC)に予想外の値が送り込まれてしまい、俗に言う「暴走」が起こってしまいます。

《第63図》ニーモニック⇔マシン語対応表

ジャンプ、コール、リターン

| × | UN COND | C | NC | Z | NZ | PE | PO | M | P | |
|------------|---------------|--------------|---------------|---------------|---------------|--------------|---------------|--------------|---------------|------------|
| JP ×, nn | C 3 n n | DA n n | D 2 n n | CA n n | C 2 n n | EA n n | E 2 n n | FA n n | F 2 n n | |
| JP ×, e | 1 8 e-2 | 3 8 e-2 | 3 0 e-2 | 2 8 e-2 | 2 0 e-2 | | | | | |
| JP (HL) | E 9 | | | | | | | | | |
| JP (IX) | DD E 9 | | | | | | | | | |
| JP (IY) | FD E 9 | | | | | | | | | |
| CALL ×, nn | CD n n | DC n n | DC n n | D 4 n n | CC n n | EC n n | E 4 n n | FC n n | F 4 n n | |
| DJNZ e | | | | | | | | | | 1 0 e-2 |
| RET × | C 9 | D 8 | D 0 | C 8 | C 0 | E 8 | E 0 | F 8 | F 0 | |
| RETI | ED 4 D | | | | | | | | | |
| RETN | ED 4 5 | | | | | | | | | |

リスタート

| | | |
|-----|-----|-----|
| RST | 00H | C 7 |
| RST | 08H | C F |
| RST | 10H | D 7 |
| RST | 18H | D F |
| RST | 20H | E 7 |
| RST | 28H | E F |
| RST | 30H | F 7 |
| RST | 38H | F F |

割り込み処理ルーチン からのリターン命令

この命令と次のリスタート命令は一般のユーザーにはあまり関係の無い命令でしょう。初心者の方や興味の無い方は、読み飛ばしてください。

一般のサブルーチンへはコール命令によってジャンプしますが、ハードウェアからの割り込み要求に対しては、割り込み処理ルーチンへのジャンプが行われます。当然、割り込み処理ルーチンからもどるためのリターン命令が必要ですが、通常の割り込み処理からもどるための命令とマスクされない割り込みに対する割り込み処理ルーチンからもどるリターン命令が用意されており、前者のニーモックは、

RETI

後者は、

RETN

となっています。

この命令実行後は、インタラプト・イネーブル・フリップ・フロップ（IFF）が、次の割り込み要求に備えてセットされ、割り込み要求が来れば、その受け入れを許可します。

リスタート命令

リスタート命令は、少しかわったサブルーチン・コール命令で、コール先のアドレス8種が最初から決められてしまっています。

コール先のアドレスを自由に指定できないコール命令なんて使い道があるのかな？ などと思われるかも知れませんが、1バイトでサブルーチン・コールが行えるのですから、省メモリのためには非常に役に立つ命令です。

ニーモニックは、

RST 00H

RST 08H

RST 10H

RST 18H

RST 20H

RST 28H

RST 30H

RST 38H

のように、コール先のアドレスに合わせて8種類が、用意されています。

リスタート命令本来の目的は、頻繁に利用するサブルーチンを、リスタート命令で呼び出せるようなプログラム構成にしておく事によって達成されますが、N-BASICでは、

RST 00H

から

RST 18H

までを、インタプリタ自身を使用し、

RST 20H

から

RST 38H

まではユーザーのために、RAM上のジャンプ・テーブルにジャンプする様に構成されています。このジャンプ・テーブルには、N-BASICがスタートした時点で、リターン命令が書き込まれますが、ユーザーが自分の利用したいサブルーチンへのジャンプ命令を書き込む事によって、好きなサブルーチンのコール用にリスタート命令を使う事ができます。

第64図にN-BASICが使うリスタート命令のコール先アドレスと、ユーザーのためのジャン

《第64図》 リスタート命令のジャンプ先

RST 00H～RST 18H

| リスタート命令 | ジャンプ先 | 使用目的 |
|---------|-------|----------------------|
| RST 00H | 0000H | N-BASICのコールド・スタート |
| RST 08H | 006AH | N-BASICのホット・スタート |
| RST 10H | 4259H | N-BASICのテキスト解釈サブルーチン |
| RST 18H | 40A6H | 各周辺機器への1バイト出力サブルーチン |

RST 20H～RST 38H

| リスタート命令 | ジャンプ先 |
|---------|-------|
| RST 20H | F1DAH |
| RST 28H | F1DDH |
| RST 30H | F1E0H |
| RST 38H | F1E3H |

プ・テーブルのアドレスを示します。

また、この命令を利用する事によって、リロケータブルなプログラムを組む事が比較的容易になり

ます。ここでは説明を省略させていただきますが、興味のある方は考えてみてください。

RST 00H

0000H 番地
へジャンプ

N-BASIC
コールド・スタート



実行前の BASIC は消去される

||

RESET

を押した状態

RST 08H

006AH 番地
へジャンプ

N-BASIC
ホット・スタート



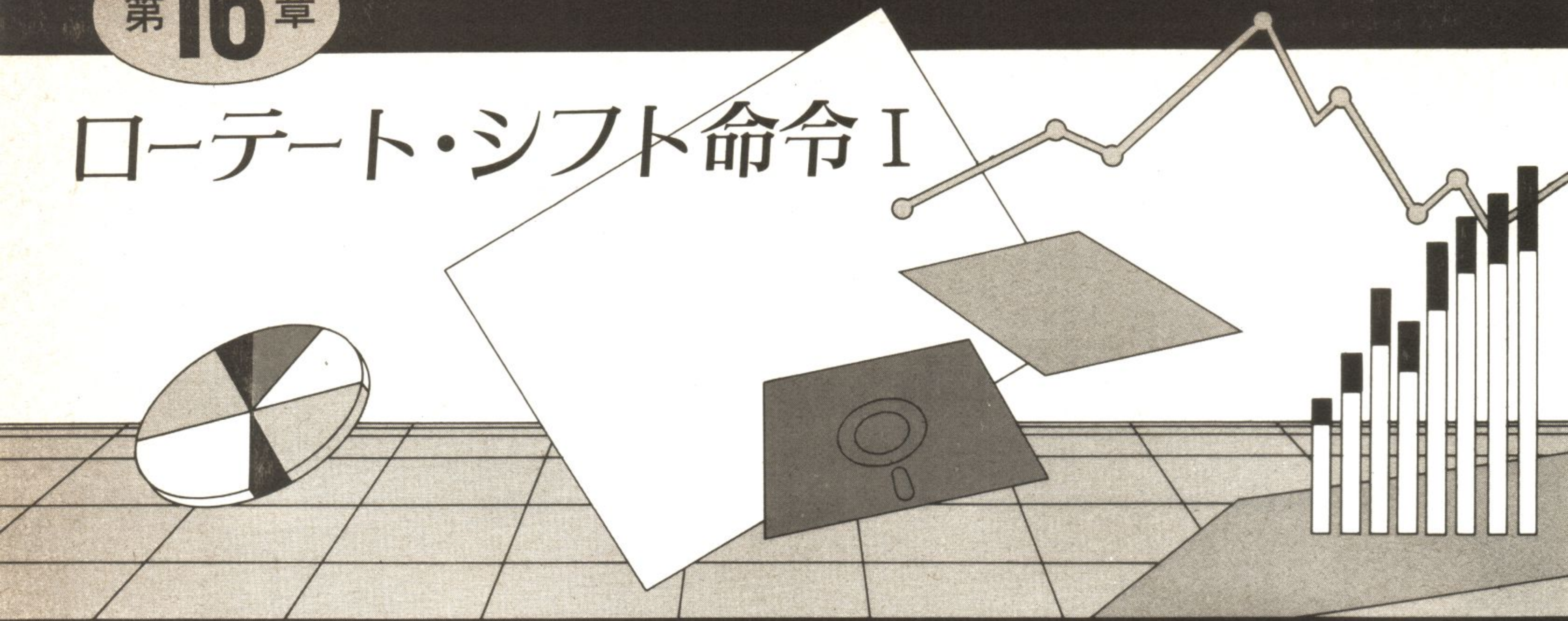
実行前のマシン語と BASIC とともに無事

||

STOP + RESET

を押した状態

ローテート・シフト命令 I



第16章から第18章に渡って、ローテート・シフト命令の説明を行います。これらの命令は、BASICなどの高級言語にはほとんど表われない、マシン語独特のもので、いろいろな応用を行って行く上で、必ず必要となって来る命令でしょう。

2進数の概念さえ理解していれば、命令自体は単純なものなのですが、マシン語で乗除算やグラフィック用のビット・データを扱う場合などの中心となるたいへん底の深い命令です。

マシン語の楽しさを十分に味わってください。

ローテート・シフトとは何か

この章で説明する、ローテート・シフト命令とは何を行うための命令なのでしょう？

皆さんも良く御存知のとおり、マイクロコンピュータのメモリには、2進数のデータが記憶されています。この2進数のデータ8ビットが集まって、1バイトを構成しているのですが、人間の側から見ると、コンピュータのメモリ上にも、

ECH

とか、

35H

のような16進数で記憶されているかのように思える事さえあります。もっとも、BASICのような

高級言語を使っている場合などは、コンピュータの中にも10進数や文字などが直接記憶されたと思っても不思議ではありません。まるで、写真のフィルムが入っていて、CRT画面に投影されている様な感じさえ受けてしまいます。

少し話がそれましたが、1バイトの16進数も実際は8ビットの2進数である事を強調したかったのです。

例をあげれば、

ECH

も、実際には、

1110 1100

であり、

35H

は、

0011 0101

であるのです。

この様な2進数の列を左右にずらす(シフトする)ための命令がシフト命令です。たとえば、

1010 1010

を1ビット、右にシフトする事によって、

→1010 1010→

〈右にシフト〉

X101 0101

と変化する事ができます。

1111 0000

を左にシフトすれば、

1110 000X

となります。Xの部分に何を入れるのかは、各種の命令が用意されていて異なります。

ローテートの場合には、2進数の列の両端をつないで輪のようにしたものを回転（ローテート）する事が目的です。

例えば、8ビットのレジスタの値を右に1ビットだけ回転（ローテート）した場合には、

```

0 0 1 0   0 0 1 0
      ▼   <右にローテート>
→ 0 0 1 0   0 0 1 0 →
      ▼
0 0 0 1   0 0 0 1

```

のようになります。

基本的なローテート ・シフト命令

Z80には、合計76個ものローテート・シフト命令が用意されていますが、その中の2個は8080当時に存在しなかった命令です。言い換えれば、8080の場合は、わずか4個のローテート・シフト命令によって全てを代用していた事になります。

この最も基本的なローテート・シフト命令は、全てAレジスタ（アキュムレータ）に働くローテート命令で、ニーモニックは、

RLCA
RRCA
RLA
RRA

の4種類で、オペランドはありません。

これらの基本命令の使い方を理解すれば、残りのほとんどの命令は、バリエーションにすぎませんから、基本命令をしっかりと理解する事が重要

です。

なお、これらの命令は、サイン・フラグ（S）及びゼロ・フラグ（Z）等には影響を与えませんが、キャリー・フラグ（CY）は重要な働きをします。

RLCA命令

RLCAは、

R o t a t e

L e f t

C i r c u l a r

A c c u m u l a t o r

の頭文字を取ったもので、Aレジスタ（アキュムレータ）の8個のビットを輪のように考えて、左へ1ビット分だけ回転（ローテート）する命令です。実行後キャリー・フラグ（CY）には、Aレジスタの第7ビットのデータであったものが送り込まれます。

例えば、Aレジスタに、

1 1 0 0 1 1 0 0

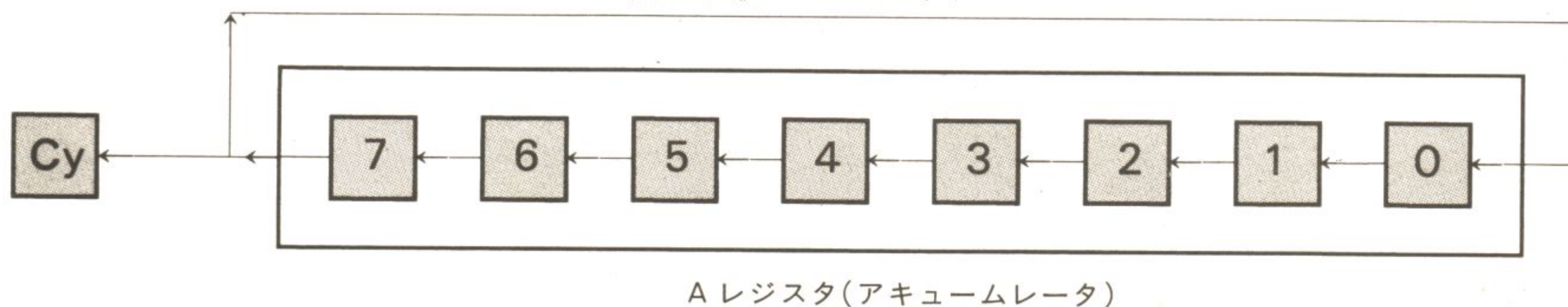
が入っている時に、この命令を実行すると、Aレジスタの値は、

1 0 0 1 1 0 0 1

に変化し、キャリー・フラグ（CY）は1にセットされます。

8080当時から用意されているローテート・シフト命令は全て1バイト命令で、この命令のマシン語は、07Hです。

《第65図》PLCA命令



Aレジスタ(アキュムレータ)

RRCA命令

RRCAは、

R o t a t e
R i g h t
C i r c u l a r
A c c u m u l a t o r

の頭文字を取ったもので、Aレジスタ（アキュムレータ）の8個のビットを輪のように考えて、

RLCA

とは逆に、右へ1ビット分回転する命令です。実行後キャリー・フラグ（CY）には、Aレジスタの第0ビットのデータであったものが送られます。

例えば、Aレジスタに、

0 0 0 1 0 0 0 1

が入っている時に、この命令を実行すると、Aレジスタの値は、

1 0 0 0 1 0 0 0

に変化し、キャリーフラグ（CY）は1にセットされます。

また、この命令のマシン・コードは0FHとなります。

RLA命令

RLAは、

R o t a t e
L e f t w i t h c a r r y
A c c u m u l a t o r

の頭文字を取ったものです。

左に、回転（ローテート）する点では、

RLCA

と同じなのですが、回転する輪が、Aレジスタにキャリー・フラグ（CY）を加えた9ビットになっている点が特色です。

例えば、キャリー・フラグ（CY）とAレジスタの内容が、

^{cy}
1 1 1 1 0 1 0 1 1

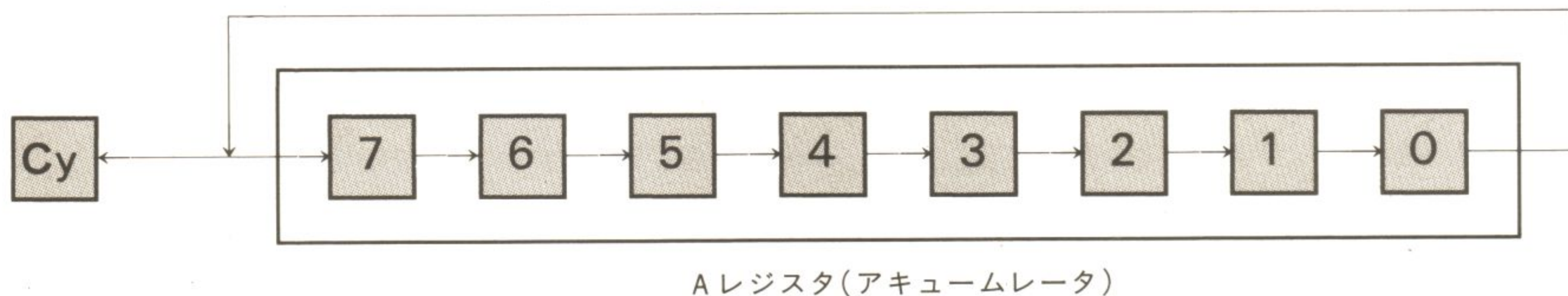
である時に、この命令を実行すると、

^{cy}
1 1 1 0 1 0 1 1 1

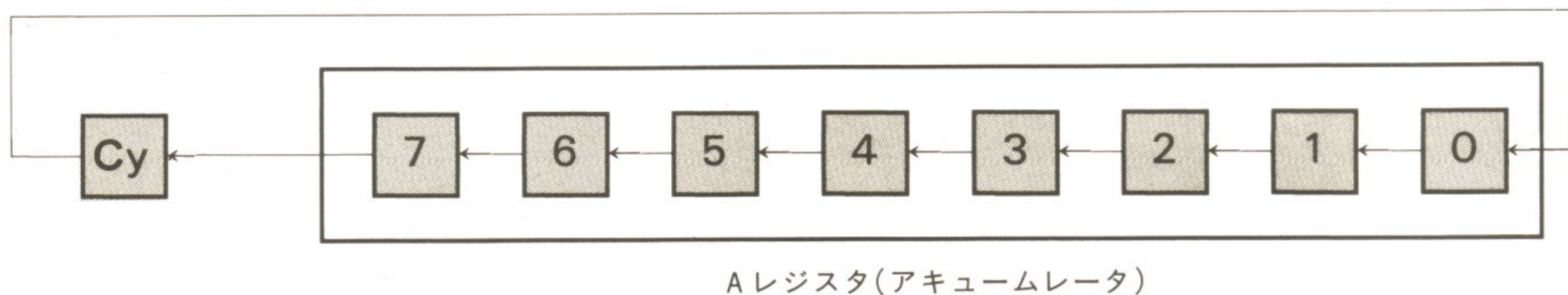
に変化します。

また、この命令のマシン・コードは、17Hとなります。

《第66図》RRCA命令



《第67図》RLA命令



RRA命令

RRAは、

R o t a t e

R i g h t w i t h c a r r y

A c c u m u l a t o r

の頭文字をとったもので、Aレジスタ（アキュムレータ）にキャリー・フラグ（CY）を加えた9ビットのデータを輪のようにして、

R L A

とは逆に、右に1ビット分回転する命令です。

例えば、キャリー・フラグ（CY）とAレジスタの内容が、

^{cy}
0 1010 0101

である時に、この命令を実行すると

^{cy}
1 0101 0010

に変化します。

又この命令のマシン・コードは、1FHとなります。

Q and A コーナー

ここで、私から読者の皆さんへ質問を出してみよう。

この問題は、FORESIGHTの会報に掲載して多くの応募をいただき、

「ぜひ、初心者向けの解説をして欲しい」との希望が多かったものです。

なお、解説と解答は第19章で紹介しますので、すこしむずかしい問題ですが、考えておいていただきたいと思います。

EXAMINATION

PC-8001のROM領域である、0000H～5FFFH番地の中から、N-BASICのプロンプト・メッセージ「Ok」がキャラクタ・コードで収納されているアドレスを見つけて「O」の収納されているアドレスを、4桁の16進数でCRT画面に表示してください。

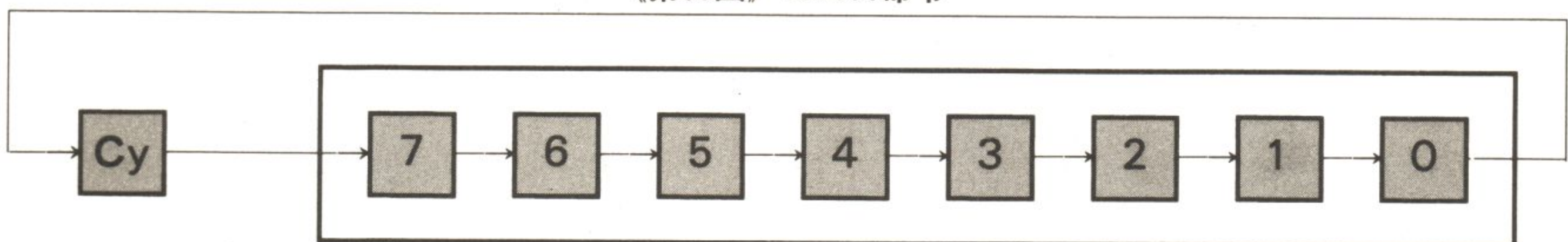
なお、5EC0H番地からのROM内サブルーチンをコールする事によって、HLレジスタ対の16進数を4桁でCRT画面に表示する事ができます。

まとめ

本章では、ローテート・シフト命令の中でも最も基本となる4個の命令について説明を行いました。実際には、この4個の命令だけでも十分に間に合うのですが、より少ないステップ数でプログラムを作るために、Z80では大幅に命令が拡張されたわけです。

さて、次章ではプログラム例でも出しながら、ローテート・シフト命令が、実際にはどのような現場で使われるのかを調べて行きたいと思います。

《第68図》 RRA命令



Aレジスタ(アキュムレータ)

8ビット乗算プログラム

第16章では、基本的な4個のローテート命令を紹介しましたが、本章では、それらの命令の応用例として、8ビットの乗算プログラムを考えてみましょう。一見して一瞬で答を算出している様に思えるコンピュータでも、実際には、人間が筆算を行う時のように1桁ずつ、計算して行く、Z80の人間らしい一面を知る事ができると思います。

なお、ここでは8ビットの乗算を行いますが、16ビットの乗算や除算なども考え方は同じです。皆さんもチャレンジしてみてください。

ローテート・シフトの目的

レジスタ内の2進数を、左右にシフトした場合の意義を考えてみましょう。

まず、だれでも考えつく事は、レジスタ内の各ビットが0であるか1であるかを、調べる事ができる点でしょう。ローテートまたはシフト命令の直後にキャリー・フラグ(CY)の値を調べれば、レジスタの最上(下)位ビットを調べる事ができます。

しかし、ローテート・シフト命令の応用範囲は、単にこれだけにはとどまらないのです。

皆さんが、10進数で表現されている数値を

10倍、100倍、1000倍～

する場合は、どのようにしますか？

まさか、123を100倍するために、

$$\begin{array}{r} 123 \\ \times 100 \\ \hline 000 \\ 000 \\ 123 \\ \hline 12300 \end{array}$$

と、筆算で求める人はいないでしょう。普通は、0を2個付け加えて、

12300

と、すぐに求められます、逆に、

10分の1、100分の1、1000分の1、～する場合にも、小数点の位置を左に移動するだけで良いわけです。

以上は、10進数の場合ですが、実際にコンピュータで扱う2進数の場合には、どうなるのかを考えてみましょう。

2進数の1を基準にして、最下位に0を付け加え、最上位のビットを削って行ってみましょう。つまり左にシフトしているのと同じ事になります。

| | | | |
|------|------|-------|----|
| 0000 | 0001 | | 1 |
| 0000 | 0010 | | 2 |
| 0000 | 0100 | | 4 |
| 0000 | 1000 | | 8 |
| 0001 | 0000 | | 16 |
| 0010 | 0000 | | 32 |

右側の10進数を見ればわかるように、最初の数値に比較して、

2倍, 4倍, 8倍, 16倍, 32倍～
となっています。

逆に, 右にシフトする事で,

2分の1, 4分の1, 8分の1～
が求められる事も, 容易に想像がつくと思います。

この様に, 2進数の数値データをnビットだけ
シフトする事は,

2^n を乗じる事
であると同時に,

小数点の位置をn個分移動する事
でもあるのです。

この性質を巧みに利用することによって, Z80
に命令として用意されていない, 乗算や除算, そ
して他の多くの関数などを実現することができる
のです。

それでは早速, このローテート・シフト命令を
利用した, 8ビット乗算プログラムを考えてしま
しょう。なお, 説明は前章で紹介した4種の基本
命令だけに限定して行います。

8ビット乗算 アルゴリズム解説

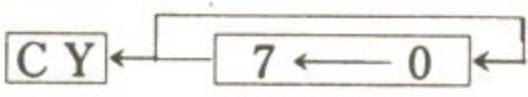
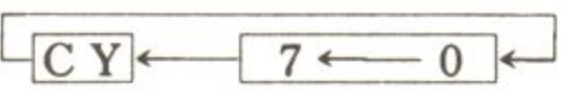
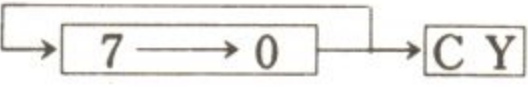
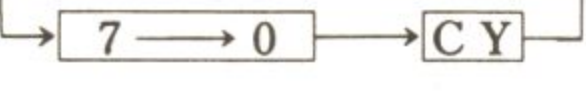
ここで紹介するプログラムは, 8ビットの乗算
を行うためのプログラムで, 基本的な4個のロー
テート・シフト命令の中で,

RRCA

RLA

の, 2個を利用しています。これらの命令の働き

《第69図》基本的なローテート・シフト命令

| ニーモック | オペレーション | フ ラ グ | | | | | |
|-------|---|-------|---|------------------|---|---|---|
| | | S | Z | H ^{P/V} | N | C | |
| RLCA |  | . | . | 0 | . | 0 | ↓ |
| RLA |  | . | . | 0 | . | 0 | ↓ |
| RRCA |  | . | . | 0 | . | 0 | ↓ |
| RRA |  | . | . | 0 | . | 0 | ↓ |

を簡単な図にしておきますので, 前章での説明を
思い出してください。(第69図)。

一般にマシン語で乗算を行う必要が生じた場合
は, どんな方法で乗算を行うのでしょうか?

私の場合には, マシン語の高速性に頼ってしま
い, ループをつくって, 加算を行う事がほとんど
です。

例えば, Bレジスタの値とCレジスタの値を掛け
合わせるために,

```

XOR  A
LOOP: ADD  A, C
      DJNZ LOOP

```

を行えば, 一応Aレジスタに積を求める事ができ
ます。ただし, Bレジスタに0を与えると, 誤動
作しますので, 事前にチェックしなければなりま
せん。

以上の方法で乗算を行うのも, プログラム自体
が簡単になって良いのですが, 合理的な方法でな
いのはたしかです。

そこで, ローテート・シフト命令を十分に活用
し, プログラムで筆算をシミュレートしてみまし
ょう。

ために,

13 * 12

を, 筆算で行ったものを, 10進数と2進数の場合
に分けて示します(第70図)。

両者を比較していただければおわかりのように,
10進数の乗算を筆算で行うためには, 最低

多桁 * 1桁

の乗算を行う事が不可欠であり,
どのように考えても, 乗算から
のがれる事はできません。私達
が小学校のころに, 九九算を記
憶しなければならなかったのも,
この様な理由によるものなので
しょう。

ところが, 2進数で筆算を行
う上では乗算は全く必要ありま
せん。なぜならば, 2進数では,
0を掛ける

か

1を掛ける

か、だけなので、結果を、

0

と、

もとの数値

のどちらかに限定する事ができるからです。

この、8ビット2進数乗算の筆算アルゴリズムをフローチャートにしたものが、第71図です。

ようするに、乗数を下位ビットから順に調べて行くと同時に、被乗数を左にシフト（2倍）して行き、乗数のビットが1であった場合のみ、シフトして来た被乗数を積に加えて行けば良いのです。

文章やフローチャートでは、少しわかりにくいかも知れませんが、アルゴリズムを箇条書きにしてみます。

1. 積を0にクリアします。
2. 乗数を右にローテート（回転）して最下位ビット（LSB）を調べます。
3. 上で調べたビットが1であれば積に被乗数を加えます。
4. 被乗数を左に1ビットシフトします。
5. 2から4までを8回くり返します。

以上のアルゴリズムによって、プログラミングを行います。例によって、被乗数、乗数および積を収納するためのメモリを決めなければなりませんが、私は次のように決定しました。

E 0 0 0 H 番地 ← 被乗数

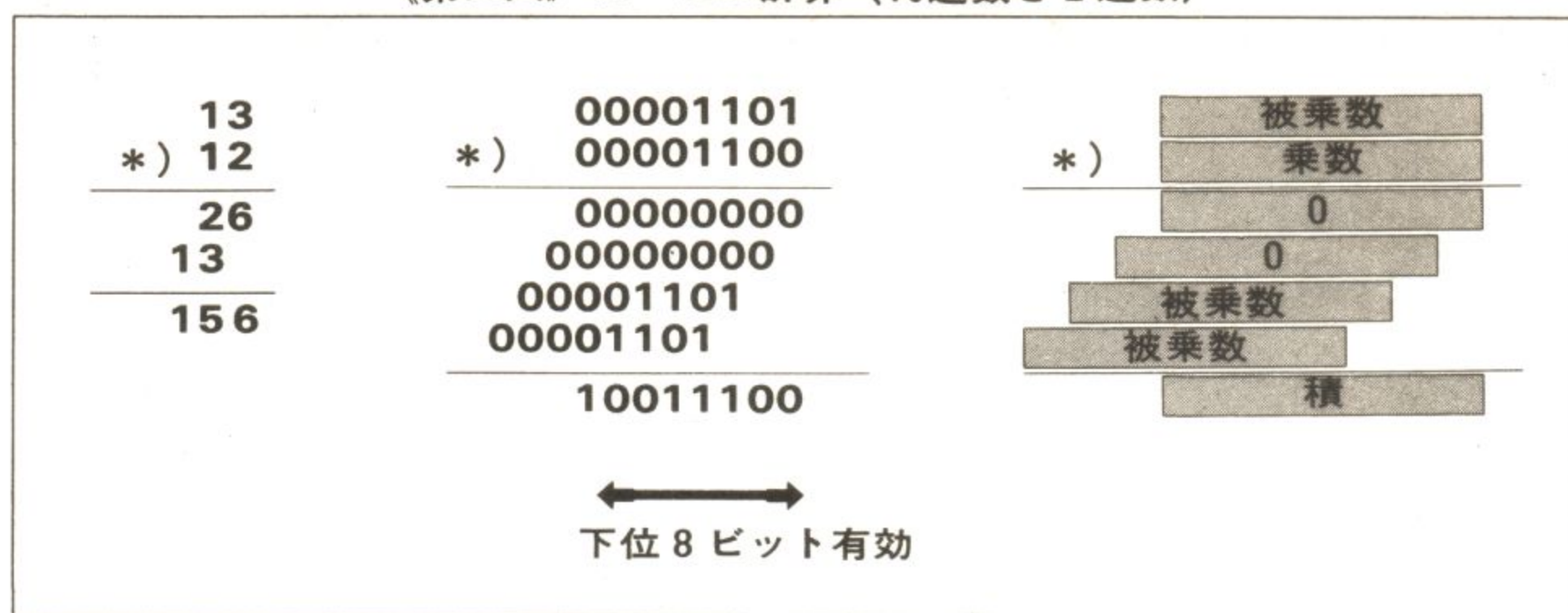
E 0 0 1 H 番地 ← 乗数

E 0 0 2 H 番地 → 積

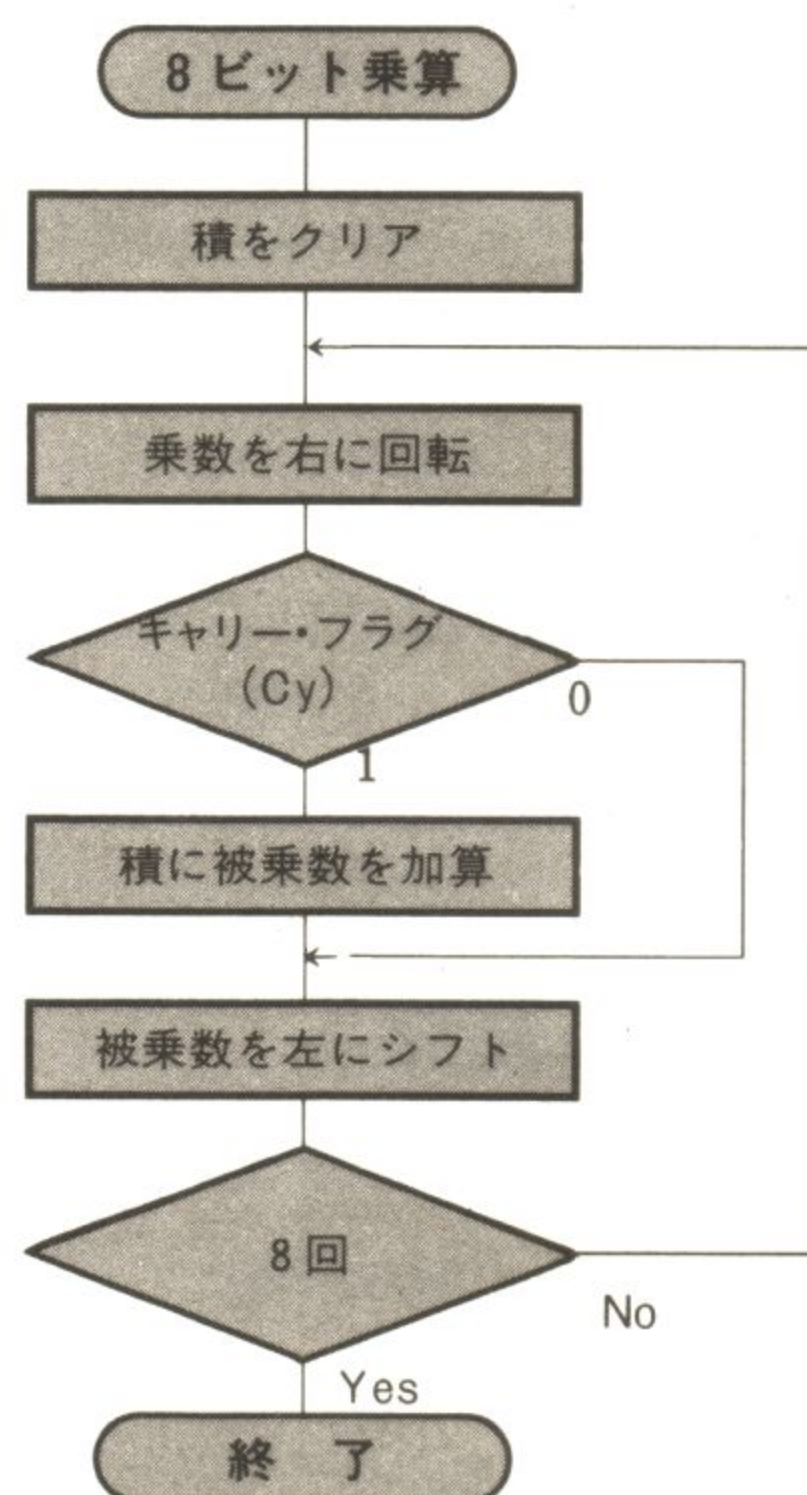
つまり、掛け合わせたい2個の数値を、E 0 0 0 H 番地とE 0 0 1 H 番地に収納してプログラムを実行するとE 0 0 2 H 番地に結果が得られるようになります。結果は、モニタのDコマンド等によって確認してください。

また、プログラムはD 0 0 0 H 番地からのマシン語としてアSEMBルしましたが、完全リロケータブルなので、好きなアドレスに置いて実行する事ができます。

データ・エリアも簡単に変更できますので、ためしてみてください。



《第71図》8ビット2進数乗算のフローチャート



8ビット乗算プログラム解説

私の考えた8ビット乗算プログラムの一例（第72図）を示します。

これまで何度もくり返して来たように、コンピュータのプログラミングで最も重要なのは、

「いかにコーディングを行うか」

ではなくて、

「いかにアルゴリズムを組立てるか」

であり、

「いかに多くのコマンドを憶えるか」

では絶対にありません。あの石井晴正先生も、

「どんなプログラム言語を用いたら良いか？」

の質問に対し、次のように答えておられます。

《第72図》 8ビット乗算プログラム

| | | |
|-----------------------------------|----------|-----------------------------------|
| ***** | | |
| 8 BIT MULTIPLICATION | | |
| ***** | | |
| ; INPUTS : (E000H) <-MULTIPLICAND | | |
| ; (E001H) <-MULTIPLIER | | |
| ; OUTPUTS : (E002H) <-PRODUCT | | |
| ; ORG 0D000H | | |
| D000 | DD2100E0 | LD IX,0E000H ; DATA START ADDRESS |
| D004 | AF | XOR A ; CLEAR |
| D005 | DD7702 | LD (IX+2), A ; PRODUCT |
| D008 | 0608 | LD B,8 ; SET LOOP COUNTER |
| D00A | DD7E01 | MULT1: LD A, (IX+1) ; |
| D00D | 0F | RRCA ; CHECK |
| D00E | DD7701 | LD (IX+1), A ; MULTIPLIER LSB |
| D011 | 3009 | JR NC, MULT2; |
| D013 | DD7E02 | LD A, (IX+2) ; |
| D016 | DD8600 | ADD A, (IX) ; ADDITION |
| D019 | DD7702 | LD (IX+2), A ; |
| D01C | DD7E00 | MULT2: LD A, (IX) ; |
| D01F | A7 | AND A ; SHIFT LEFT |
| D020 | 17 | RLA ; MULTIPLICAND |
| D021 | DD7700 | LD (IX), A ; |
| D024 | 10E4 | DJNZ MULT1 ; LOOP 8 TIMES |
| D026 | C3665C | JP 5C66H ; JUMP TO MONITOR |

「アルゴリズムさえ正しければ、どのような言語を用いても正しい結果が得られる。ただ言語によって異なるのは実行速度のみである」と。全く先生の話されたとおりだと思います。もっとも、私の様な凡人にとっては、プロミグラングに要する手間や時間も重要な要素となって来ますが、私のまわりにはBASICよりマシン語の方が手間がかからないと言っている友人も多く、驚いてしまいます。

プログラムのコーディングにあたっては、普段あまり使用される機会の無いインデックス・レジスタ (IX) を多用してみました。そのため、実行速度やメモリ容量の点では多少不利になっていますが、Z80らしい見易いプログラムになっていると思います。

8回のループを行うためのループ・カウンタとしては、一般的なBレジスタを利用しています。

アセンブル・リストには、コメントとして、アルゴリズムを入れておきましたので、フローチャートと共に、ニーモニックを追ってみてください。少し複雑なプログラムなので多少解析しづらいか

かも知れません。

なお、このプログラムでは、Aレジスタのデータを左に1ビットだけシフトするために、

AND A
RLA

を行っています。これは、論理演算命令を実行してキャリー・フラグ (CY) を0にリセットした後、ローテートする事によって、シフトを実現しているものです。(第73図)。

この時に、

AND A

を行わないと、ローテートした時点で、Aレジスタの最下位ビット (LSB) に1が入ってしまいます。

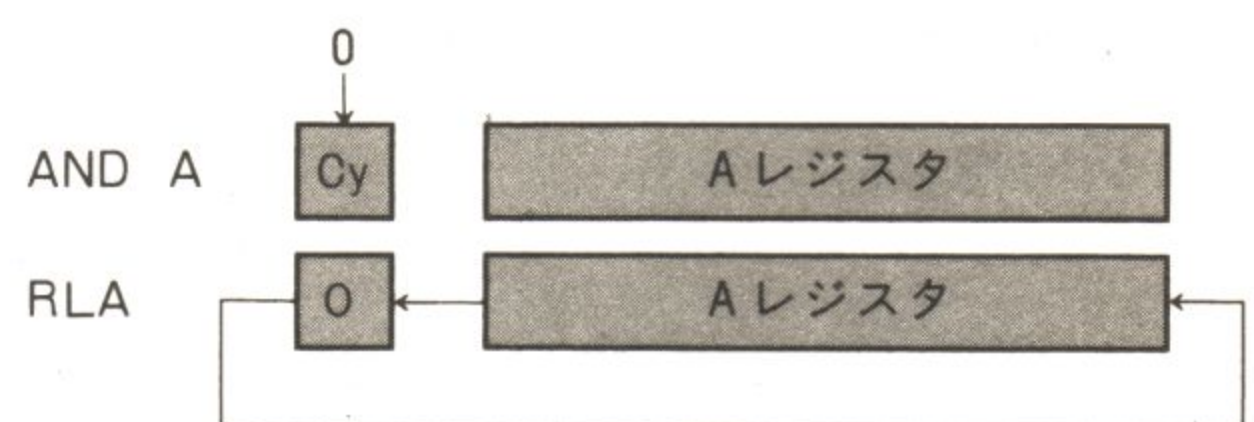
実際に、Aレジスタのデータを左にシフトするだけであれば、

ADD A, A

を行って、Aレジスタを2倍すれば良いのですが、

ここでは、ローテート命令を使いたかったので、この方法をとりました。

《第73図》Aレジスタのデータを左に1ビットシフト

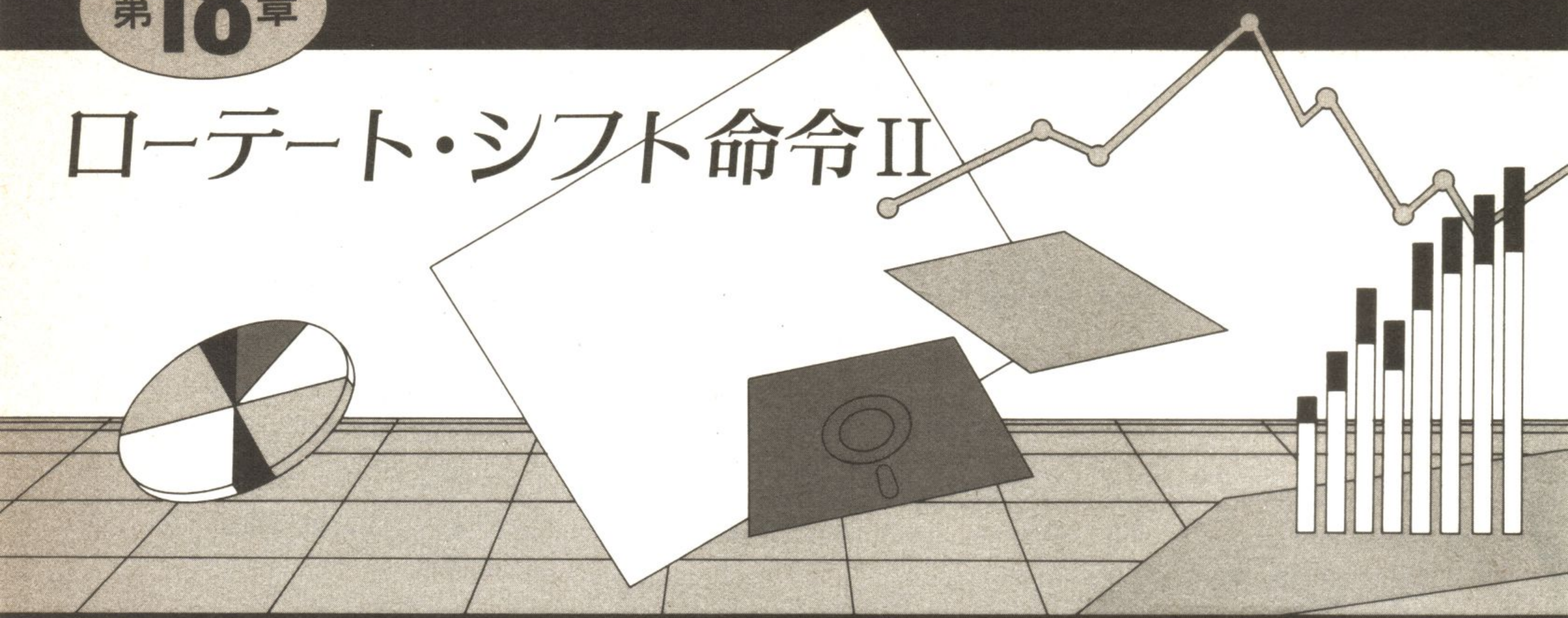


まとめ

第16章から第17章に渡って基本的な4個のローテート・シフト命令を説明しましたが、いかがだったでしょうか？ 私はこれらの命令が最もマシン語的な感覚の命令ではないかと思っています。

さて、次の章では、Z80になって拡張された多くのローテート・シフト命令を簡単に紹介したいと思います。

ローテート・シフト命令II



本章では8080からZ80へ拡張された時点で追加されたローテート・シフト命令を、ざっとですが説明したいと思います。

何度もくり返して述べて来た様に、これらの追加された命令は、あくまで

「あれば便利な命令」
に過ぎず、

「必要不可欠な命令」
ではありません。Z80を使いこなそうと思えば重要な命令かもしれませんが、まず、基本的な4種の命令

RLCA
RRCA
RLA
RRA

だけでも十分に熟知しておいて欲しいものです。

ですから、本章の説明も上記の4種の命令を補足する様な気軽なつもりで読んでいただきたいと思います。数多くのニーモニックが用意されていますから、その中の一つや二つ忘れてたり区別がつけられなくてもプログラミングの大勢には、ほとんど影響ありません。

RLC命令

基本命令の

RLCA

が、Aレジスタのみに有効であったものを、他のレジスタやメモリ上の値に対しても有効にしたもので、

RLC L

RLC (HL)

RLC (IX+12H)

などの様にして使います。

ちなみに、

RLC A

は、基本命令の

RLCA

と同じ働きを行いますが、フラグへの影響が異なり、たとえば基本命令ではサイン・フラグ(S)やゼロ・フラグ(Z)に対して全く影響を及ぼさなかったのに対してZ80で拡張されたローテート・シフト命令では、ほとんどのフラグに対して影響を与える様になっています。また、アセンブルした場合にも2バイト命令になります。

RL命令

基本命令の

RLA

を、Aレジスタ以外のレジスタやメモリ上の値に対して直接働く様にしましたもので、

RL B

RL (IY)

RL (IY+127)

などの様にして使います。

RL A

は、基本命令の

RLA

と同じ働きを行いますが、やはりフラグへの影響は異なります。

RRC命令

基本命令の

RRC A

を、Aレジスタ以外のレジスタやメモリ上の値に対して直接働く様にしましたもので、

RRC E

RRC (HL)

RRC (IX)

などの様にして使います。

RLC A

は基本命令の

RRC A

と同じ働きを行いますが、やはりフラグへの影響は異なります。

RR命令

基本命令の

RR A

を、Aレジスタ以外のレジスタやメモリ上の値に対して直接働く様にしましたもので、

RR C

RR (HL)

RR (IX-12H)

のようにして使います。

RR A

は、基本命令の

RR A

と同じ働きを行いますが、やはりフラグへの影響は異なります。

SLA命令

SLAは、

Shift

Left

Arithmetic

の頭文字をとったもので、

「数学的な左シフト」

の意味です。

具体的には、もとの8ビットの情報を左に1ビット分シフトし、最上位ビットからの情報をキャリー・フラグ(CY)にセットし、最下位ビットを無条件に0にしてしまいます。Aレジスタに対する

SLA A

を、今までに説明した他の命令に置き換えれば、

AND A

RL A

の2ステップで代用する事ができますが、1ステップ目は、キャリー・フラグ(CY)を0にリセットするために実行する命令です。

SRA命令

SRAは、

Shift

Right

Arithmetic

の頭文字をとったもので、

「数学的な右シフト」

の意味です。

具体的には、もとの8ビットの情報を右に1ビット分シフトし、最下位ビットからの情報をキャリー・フラグ (CY) にセットしますが、最上位ビット (第7ビット) は全く変化しません。

この命令によって、正負を考慮した1バイトの数値 (-128~127) を1ステップで2分の1にする事ができます。

実際には、

SRA C
SRA (HL)
SRA (IY+00H)

の様に使用します。

SRL命令

SRLは、

Shift

Right

Logical

の頭文字をとったもので、

「論理的な右シフト」

の意味です。

具体的には、もとの8ビットの情報を右に1ビット分シフトし、最下位ビットからの情報をキャリー・フラグ (CY) にセットし、最上位ビットを無条件に0にしてしまいます。Aレジスタに対する

SRL A

を、今までに説明した他の命令に置き換えれば、

AND A

RR A

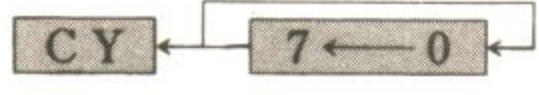



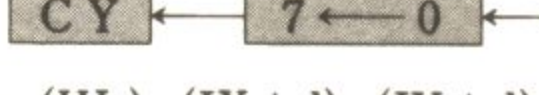
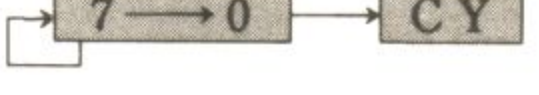



の2ステップで代用する事ができます。

先程紹介しました、

SRA

と逆の命令と考える事ができます。

《第74図》Z80で拡張されたローテート・シフト命令群

| ニーモニック | オペレーション | フ ラ グ SZHPNC |
|-------------------------------|--|----------------------------|
| PLC r PLC(HL) PLC(IX+d) |  | ↑↑0P0↑ ↑↑0P0↑ ↑↑0P0↑ |
| RLC(IY+d) | r, (HL), (IX+d), (IY+d) | ↑↑0P0↑ |
| RL r RL(HL) RL(IX+d) |  | ↑↑0P0↑ ↑↑0P0↑ ↑↑0P0↑ |
| RL(IY+d) | r, (HL), (IX+d), (IY+d) | ↑↑0P0↑ |
| RRC r RRC(HL) RRC(IX+d) |  | ↑↑0P0↑ ↑↑0P0↑ ↑↑0P0↑ |
| RRC(IY+d) | r, (HL), (IX+d), (IY+d) | ↑↑0P0↑ |
| RR r RR(HL) RR(IX+d) |  | ↑↑0P0↑ ↑↑0P0↑ ↑↑0P0↑ |
| RR(IY+d) | r, (HL), (IX+d), (IY+d) | ↑↑0P0↑ |
| SLA r SLA(HL) SLA(IX+d) |  | ↑↑0P0↑ ↑↑0P0↑ ↑↑0P0↑ |
| SLA(IY+d) | r, (HL), (IX+d), (IY+d) | ↑↑0P0↑ |
| SRA r SRA(HL) SRA(IX+d) |  | ↑↑0P0↑ ↑↑0P0↑ ↑↑0P0↑ |
| SRA(IY+d) | r, (HL), (IX+d), (IY+d) | ↑↑0P0↑ |
| SRL r SRL(HL) SRL(IX+d) |  | ↑↑0P0↑ ↑↑0P0↑ ↑↑0P0↑ |
| SRL(IY+d) | r, (HL), (IY+d), (IY+d) | ↑↑0P0↑ |
| RLD | A  (HL) | ↑↑0P0· |
| RRD | A  (HL) | ↑↑0P0· |

RLD命令

RLDは、
Rotate
Left
Digit

の頭文字をとったものでオペランドはありません。

今までに説明して来たローテート・シフト命令は全て、1ビット単位で動作を行いましたが、この命令では、4ビット単位でのローテートを行う事ができます。ローテートの対象となるのは、HLレジスタ対で指定するメモリ上のデータで、Aレジスタも重要な役目を持っています。

具体的には説明するよりも第74図のオペレーションの項を見ていただいた方が早いでしょう。

実際には、メモリ上にBCDコードで収納されている10進数を桁単位でシフトする場合に便利なのですが、私自身が一度も利用した事の無い事を考えるとあまり一般的な命令では無いのかも知れません。

RRD命令

RRDは、
Rotate
Right
Digit

の頭文字をとったもので、やはりオペランドはありません。

RLD
のローテート方向を逆にしたものと考えれば良いと思います。

BCDコードについて

命令の説明文中にBCDコードという少し聞き慣れない単語が登場して来ました。このBCDコードを説明する前に、いくつか「コード」と付く単語を並べてみましょう。

バイナリ・コード (BINARY CODE)

16進コード (HEXA CODE)

アスキー・コード (ASCII CODE)

キャラクタ・コード

(CHARACTER CODE)

《第75図》ローテート・シフト命令 (ニーモニック→マシン語対応表)

| × | A | B | C | D | E | H | L | (HL) | (IX +d) | (IY +d) |
|-------|----------|----------|----------|----------|----------|----------|----------|----------|-----------------|-----------------|
| RLC × | CB 07 | CB 00 | CB 01 | CB 02 | CB 03 | CB 04 | CB 05 | CB 06 | DD CB d06 | FD CB d06 |
| RRC × | CB 0F | CB 08 | CB 09 | CB 0A | CB 0B | CB 0C | CB 0D | CB 0E | DD CB d0E | FD CB d1E |
| RL × | CB 17 | CB 10 | CB 11 | CB 12 | CB 13 | CB 14 | CB 15 | CB 16 | DD CB d16 | FD CB d06 |
| RR × | CB 1F | CB 18 | CB 19 | CB 1A | CB 1B | CB 1C | CB 1D | CB 1E | DD CB d1E | FD CB d1E |
| SLA × | CB 27 | CB 20 | CB 21 | CB 22 | CB 23 | CB 24 | CB 25 | CB 26 | DD CB d26 | FD DB d26 |
| SRA × | CB 2F | CB 28 | CB 29 | CB 2A | CB 2B | CB 2C | CB 2D | CB 2E | DD CB d2E | FD CB d2E |
| SRL × | CB 3A | CB 38 | CB 39 | CB 3A | CB 3B | CB 3C | CB 3D | CB 3E | DD CB d3E | FD CB d3E |
| RLD | | | | | | | | ED 6F | | |
| RRD | | | | | | | | ED 67 | | |

| | A |
|------|----|
| RLCA | 07 |
| RRCA | 0F |
| RLA | 17 |
| RRA | 1F |

BASIC等の中間コード

etc.

これらをながめているうちに、「コード」と付けられている単語の共通点に気が付いた方もおられると思います。その共通点とは、

「コンピュータのメモリへの収納方法」を表わしている事です。

例えば、中間コードは、BASIC等の命令のメモリへの収納方法の一つであり、アスキー・コードやキャラクタ・コードはコンピュータで文字を扱う場合に使用されます。

さて、問題のBCDコードですが、これは数値を扱う場合に利用されるもので、人間が通常扱っている10進数を簡単にメモリ上で扱うために考えられた方法です。

具体的には、第76図を見ていただければわかる様に1バイトを4ビットずつに分けて、2桁の10進数を表わすものです。

たとえば、10進数の12は、16進数では0CHと表現しますが、BCDコードでは12Hとなります。

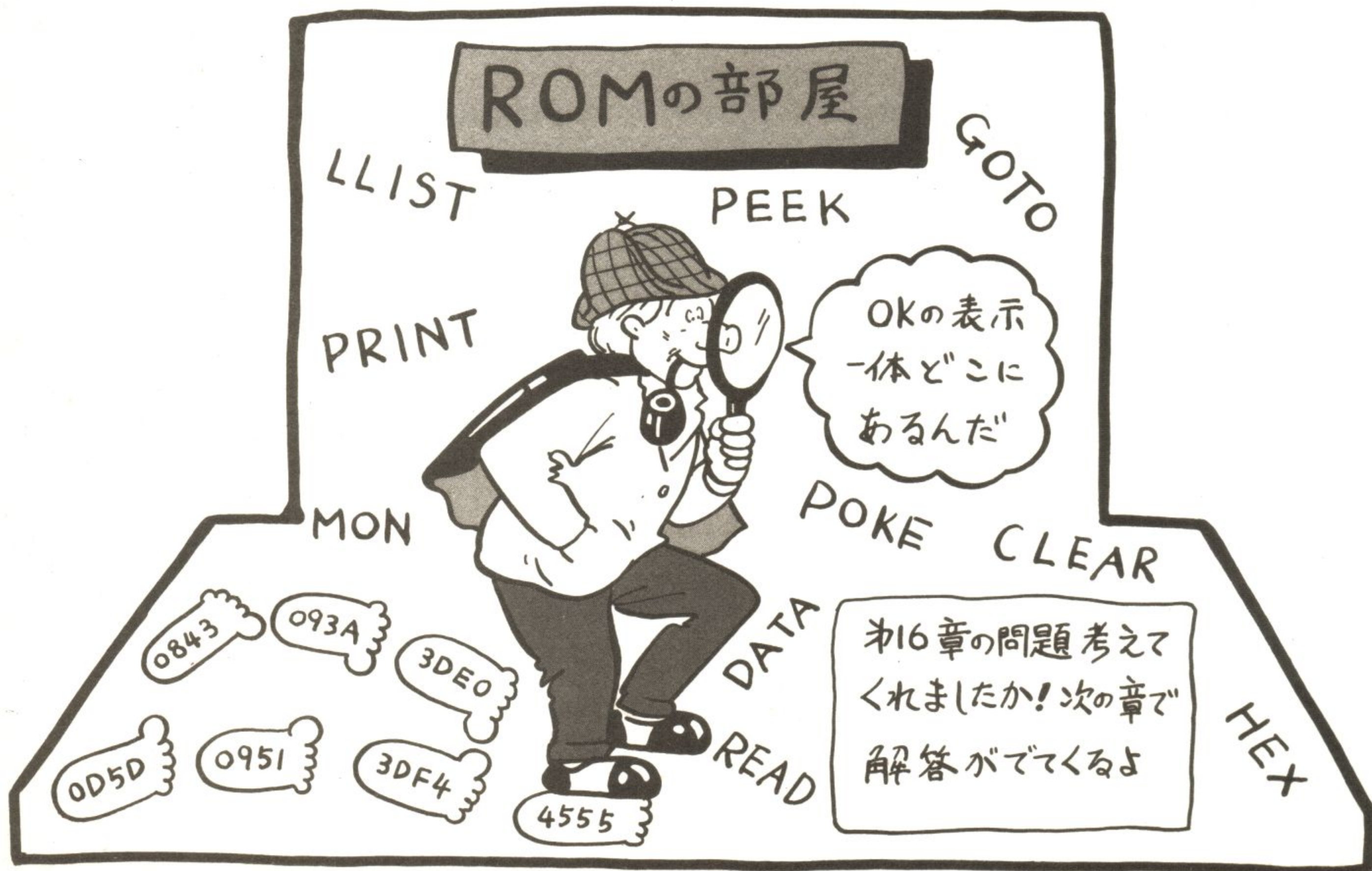
この様に、BCDコードで表現されたデータを

《第76図》BCDコードの表現

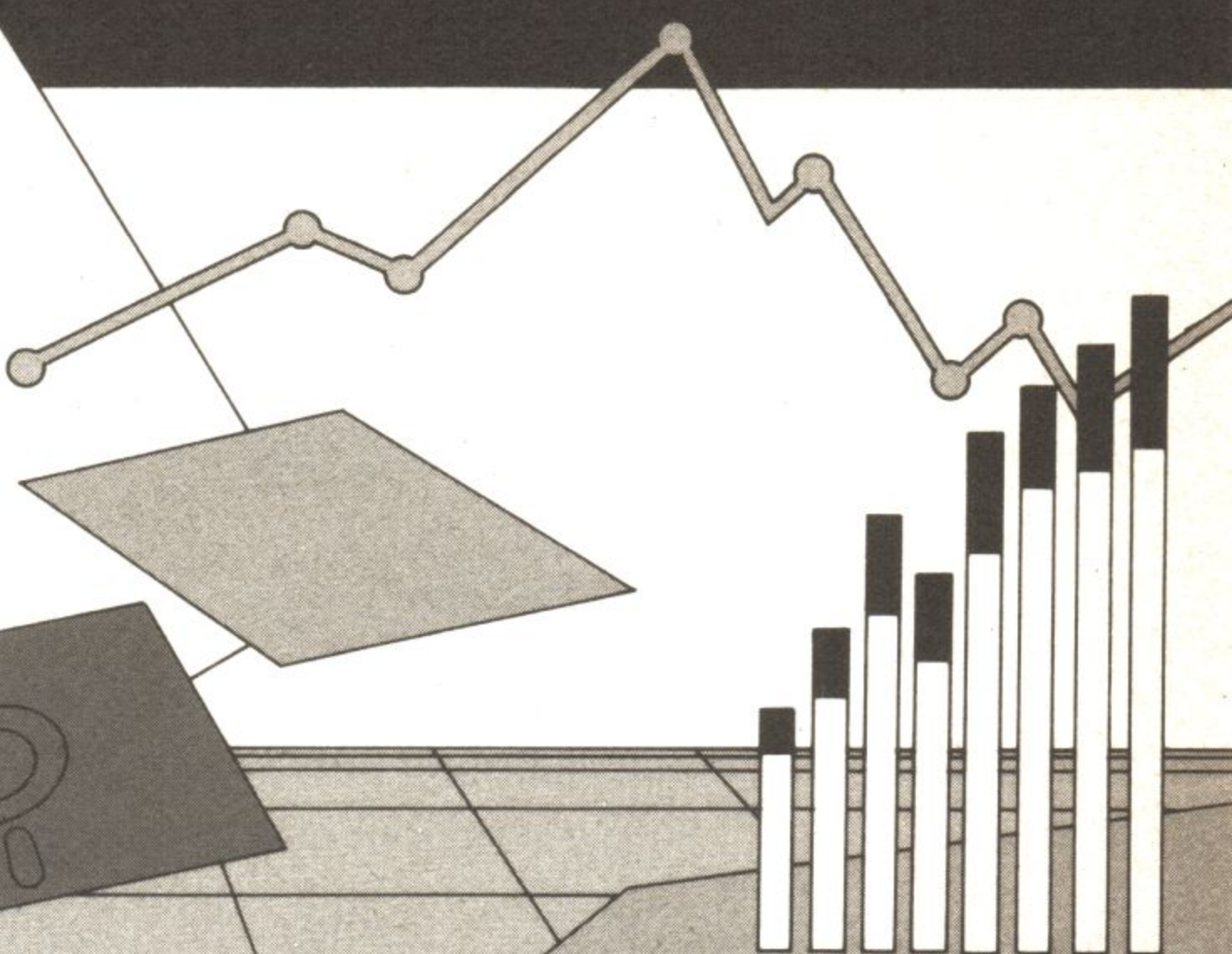
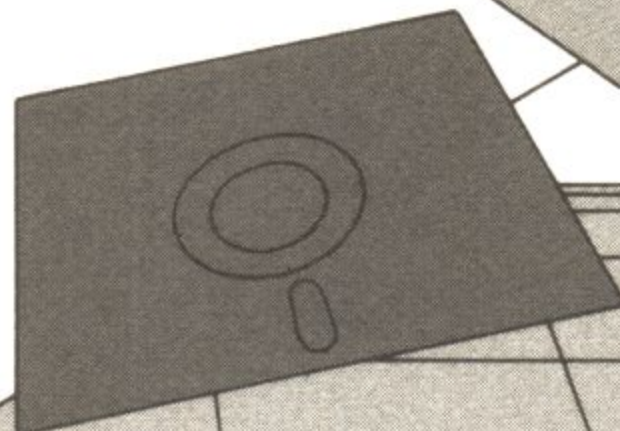
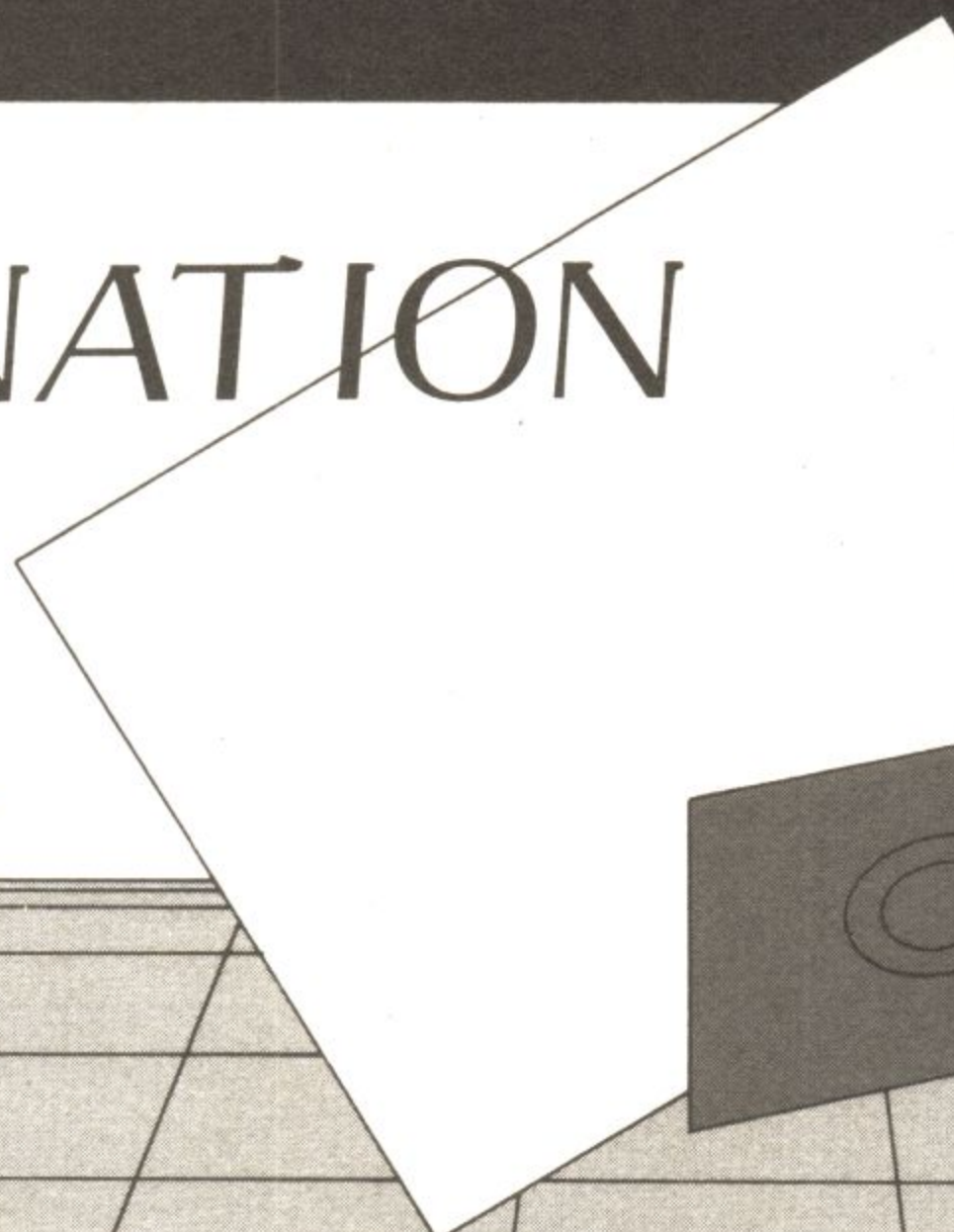
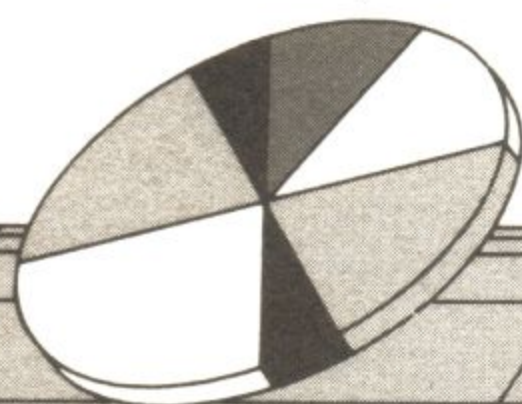
| 10進数表現 | 16進数表現 | BCDコード表現 |
|--------|--------|----------|
| 0 | 00 | 00 |
| 1 | 01 | 01 |
| 2 | 02 | 02 |
| 3 | 03 | 03 |
| 4 | 04 | 04 |
| 5 | 05 | 05 |
| 6 | 06 | 06 |
| 7 | 07 | 07 |
| 8 | 08 | 08 |
| 9 | 09 | 09 |
| 10 | 0A | 10 |
| 11 | 0B | 11 |
| 12 | 0C | 12 |
| 13 | 0D | 13 |
| 14 | 0E | 14 |
| 15 | 0F | 15 |
| 16 | 10 | 16 |
| 17 | 11 | 17 |
| 18 | 12 | 18 |
| } | } | } |

16進でダンプする事によって、人間の方は簡単に10進数のイメージとして受け取る事ができます。

1バイトで2桁の10進数しか表現できませんので、省メモリを必要とする場合などには不向きですが、人間の方から考えれば、最も自然に感じる事のできる無理のない表現方法と言えましょう。



EXAMINATION



EXAMINATION

PC-8001のROM領域である、0000H~5FFFH番地の中から、N-BASICのプロンプト・メッセージ「Ok」がキャラクタ・コードで収納されているアドレスを見つけだして「O」の収納されているアドレスを、4桁の16進数でCRT画面に表示して下さい。

なお、5EC0H番地からのROM内サブルーチンをコールする事によって、HLレジスタ対の16進数を4桁でCRT画面に表示する事ができます。

上記の問題は、第16章の最後で私から読者の皆さんに出題したものです。

考えられた方は、以外に面倒なプログラムなので驚かれたのではないででしょうか。それとも、あまりの安易さに、考える気力さえ起きなかったのかも知れません。

いずれにしても、この辺で少しの間命令の紹介を休ませていただき、私なりの解答を示しておきたいと思います。皆さんもお付き合いください。

ただし、ここで示すいくつかのプログラムはあくまでも参考例として考えてください。皆さんが御自分でつくったプログラムが、模範解答である事を期待しています

私の解答例

まず最初に、私が出題した時点で考えておいた解答例を第77図に示します。

このプログラムは、「Ok」の文字列をさがし出す事のみを目的として、できる限り少ないステップにしようと思い組んだものです。当然の事ですが、今までに説明した命令のみで成り立ったプログラムです。

プログラム自体は非常に簡単なものですが、一応順を追って説明して行きます。

メモリを指定するためのポインタとしては、HLレジスタ対を使い、5FFFH番地から0000H番地に向かって調べて行きます。メモリ上を、わざわざ逆に調べて行くのは、プロンプトをさがし出した時点で「k」ではなく「O」の収納されているアドレスを表示しなければならないので、その時点でポインタをもどすのが無駄だと思ったからです。

また、注意しなければならないのは、16ビットのディクリメント命令ではフラグに影響を及ぼさない点です。つまり、第77図で最初の

```
JR      NZ, LOOP
```

は、直前の

```
DEC     HL
```

に対する条件ジャンプでは無く、もう1ステップ前の

CP 'k'

に対する条件判断なのです。中途半端な場所に

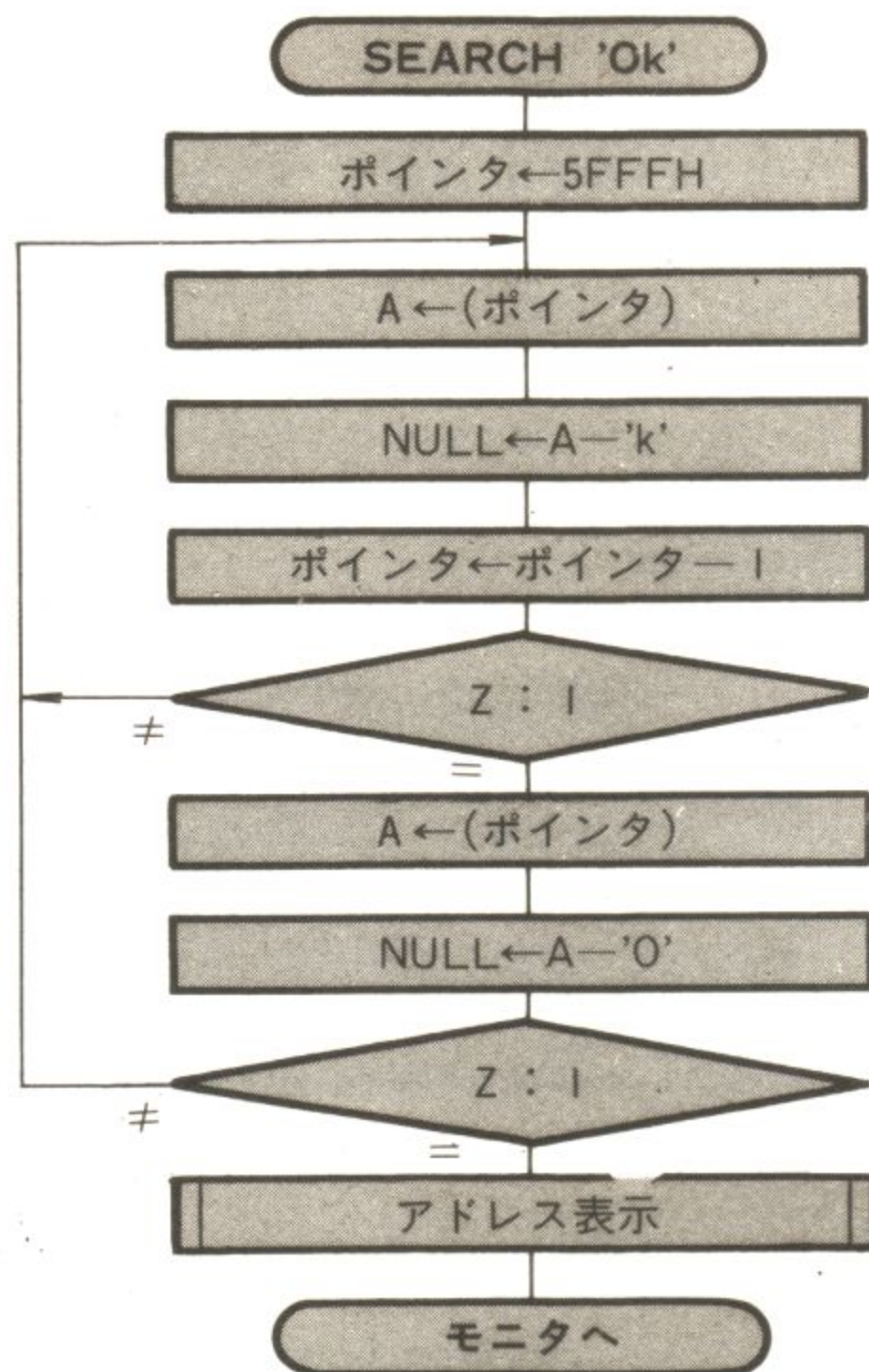
DEC HL

が入っているため、わかりにくいプログラムになっていますが、ステップ数の軽減には役立っています。

なお、第78図のプログラムでは、プロンプトを見つけた段階で、アドレスを16進4桁で表示した後モニタにジャンプしてしまいます。

本当なら、24KバイトのROM領域内に「Ok」のキャラクタ・コードが並んでいる場所が1箇所のみとは限らず、後で登場する第81図のようなプログラムを組む事が望ましいのですが、本書が入門書である事も考慮し、偶然ROM領域内に1組の「Ok」しか存在しない事が確認済みであったため、このようなプログラムにしました。もちろん、余力のある方は別の方法を考えてみてください。

《第78図》 解答例1のフローチャート



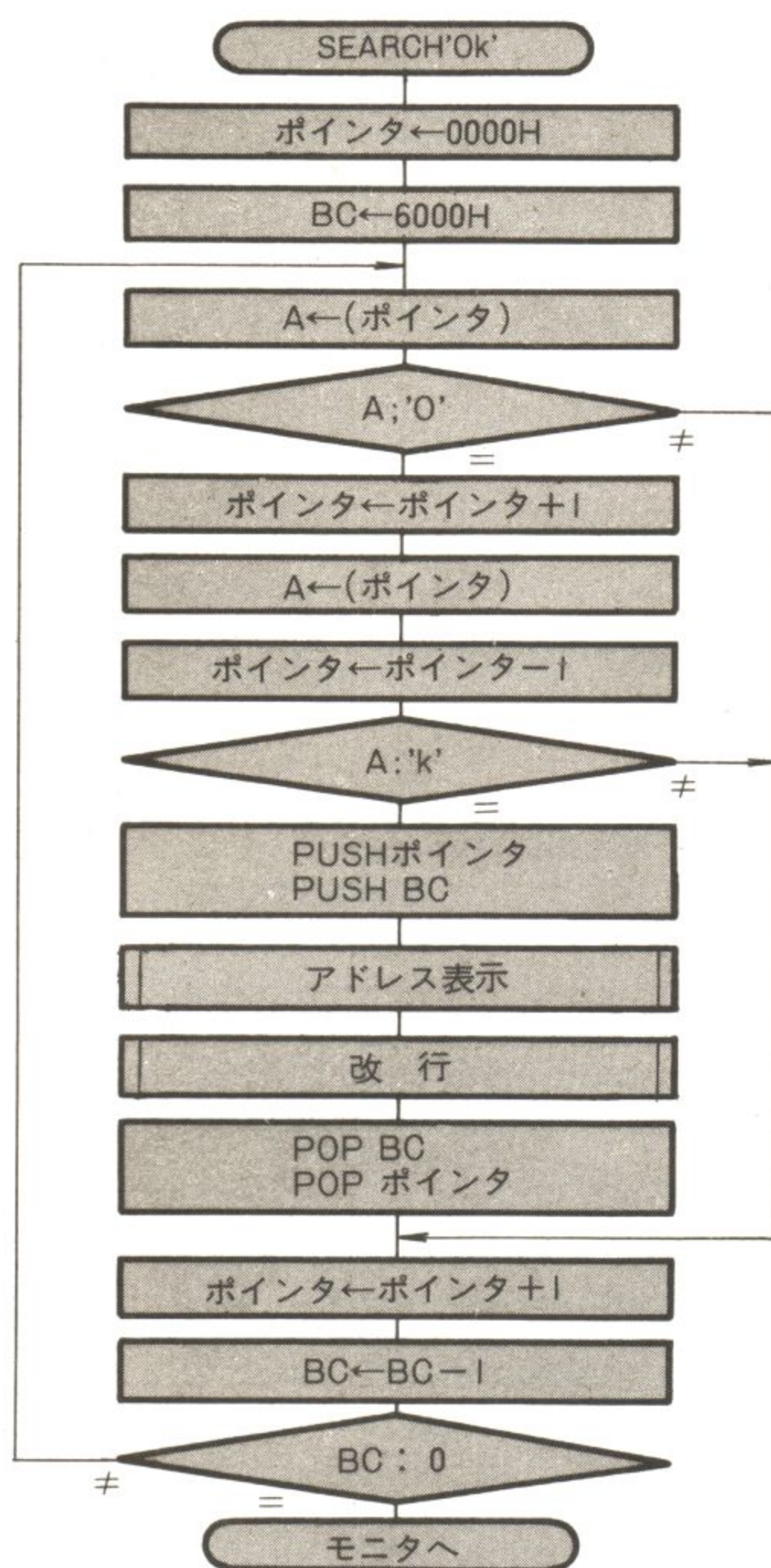
《第77図》 解答例1

| | | |
|------|--------|------------------|
| | | ***** |
| | | SEARCH 'Ok' |
| | | ***** |
| | | ; |
| | | ORG 0E000H |
| | | ; |
| E000 | 21FF5F | LD : HL, 5FFFH |
| | | ; |
| E003 | 7E | LOOP: LD A, (HL) |
| E004 | FE6B | CP 'k' |
| E006 | 2B | DEC HL |
| E007 | 20FA | JR NZ, LOOP |
| E009 | 7E | LD A, (HL) |
| E00A | FE4F | CP 'O' |
| E00C | 20F5 | JR NZ, LOOP |
| E00E | CDC05E | CALL 5EC0H |
| E011 | C3665C | JP 5C66H |

《第79図》 解答例2

| | | |
|------|--------|------------------|
| | | ***** |
| | | SEARCH 'Ok' |
| | | ***** |
| | | ; |
| | | ORG 0E000H |
| | | ; |
| E000 | 210000 | LD HL, 0000H |
| E003 | 010060 | LD BC, 6000H |
| | | ; |
| E006 | 7E | LOOP: LD A, (HL) |
| E007 | FE4F | CP 'O' |
| E009 | 2011 | JR NZ, NEXT |
| E00B | 23 | INC HL |
| E00C | 7E | LD A, (HL) |
| E00D | 2B | DEC HL |
| E00E | FE6B | CP 'k' |
| E010 | 200A | JR NZ, NEXT |
| E012 | E5 | PUSH HL |
| E013 | C5 | PUSH BC |
| E014 | CDC05E | CALL 5EC0H |
| E017 | CDCA5F | CALL 5FCAH |
| E01A | C1 | POP BC |
| E01B | E1 | POP HL |
| | | ; |
| E01C | 23 | NEXT: INC HL |
| E01D | 0B | DEC BC |
| E01E | 7B | LD A, B |
| E01F | B1 | OR C |
| E020 | 20E4 | JR NZ, LOOP |
| E022 | C3665C | JP 5C66H |

《第80図》 解答例 2 のフローチャート



《第81図》 解答例 3

```

; *****
;      SEARCH  'Ok'
; *****
;      ORG    0E000H
;
E000 210000      LD    HL, 0000H
;
E003 110060      IFEND: LD    DE, 6000H
E006 CDD35E      CALL  5ED3H
E009 D2665C      JP     NC, 5C66H
;
E00C E5          PUSH  HL
E00D 012AE0      LD     BC, TABLE
;
E010 0A          LOOP: LD     A, (BC)
E011 A7          AND    A
E012 2B07        JR     Z, NEXT
E014 BE          CP     (HL)
E015 2004        JR     NZ, NEXT
E017 03          INC    BC
E018 23          INC    HL
E019 1BF5        JR     LOOP
;
E01B D1          NEXT: POP    DE
E01C 23          INC    HL
E01D 20E4        JR     NZ, IFEND
E01F E5          PUSH  HL
E020 EB          EX     DE, HL
E021 CDC05E      CALL  5EC0H
E024 CDCA5F      CALL  5FCAH
E027 E1          POP    HL
E028 1BD9        JR     IFEND
;
E02A 4F6BFC0D    TABLE: DB    'Ok', 0FCH, 0DH, 0AH, 0
E02E 0A00

```

もう一つの解答例

東大阪市の野本正司さんから（月刊マイコン連載中に）、たいへんすばらしい解答例と解説をいただきました。まさに、

「我が意を得たり」

というような内容でしたので、ここに発表させていただきます。

なお、誌面の都合などから、文章の一部を変更し、フローチャート等勝手に付けさせていただきました。あらかじめ御了承ください。

野本さんからの手紙

最初にできたのが第79図です。「O」と「k」のキャラクタ・コード 4FH, 6BH をさがす簡単なものですが、ハンド・アセンブルした後重大な問題に気がつきました。すなわち第79図では、N-BASIC のプロンプト以外にも反応するのです。たとえば、データやマシン語などが偶然、4FH, 6BH と並んでいる場合にも反応してしまうのです。

そこで考えたのが第81図です。

プロンプトは 52EDH 番地からの文字列表示サブルーチンによって表示されるはずですから、エンド・マーク 00H があり、更に改行を行うために、0DH, 0AH が含まれているはずで

普通なら、ここまでで終わりですが、HAL研究所のPCG-8100を使用中、プロンプト「OK」の直後に1キャラクタの不特定パターンが、表示される事に気がつきました。調べた結果、キャラクタ・コードがFCHのキャラクタである事が判明したのです。

以上の考察からN-BASICのプロンプトは、4FH, 6BH, FCH, ODH, OAH, OOHから構成されていると考えられ、単に、「OK」のキャラクタ・コード2バイトをさがしただけでは不完全な事がわかります。

第81図では、第79図のように、4FH, 6BHを拾い出して調べるのではなく、調べるデータをあらかじめテーブルに用意しておいて、そのデータが全て一致するところを見つけ出します。ちょうど、N-BASIC等のINSTR関数みたいです。

LOOPというラベルのついているブロックは、サーチすべきデータのエンド・マークOOHまですべて合っていればゼロ・フラグ(Z)をたて、途中でひとつでも異なっていればゼロ・フラグ(Z)をたおしてNEXTのラベルにジャンプします。

まず、第79図の方です。さすがにマシン語だけあって瞬時にして終わりました。24Kバイトもの領域をまさに一瞬です。

第79図の実行結果表示されたのは以外にも3B60だけでした。念のため3B60H番地からタンブしてみると、4FH, 6BH, FCH, ODH, OAH, OOH, に間違いありませんでした。

24KバイトものROM領域に「O」と「K」のペアは、たった1組だけだったのです。

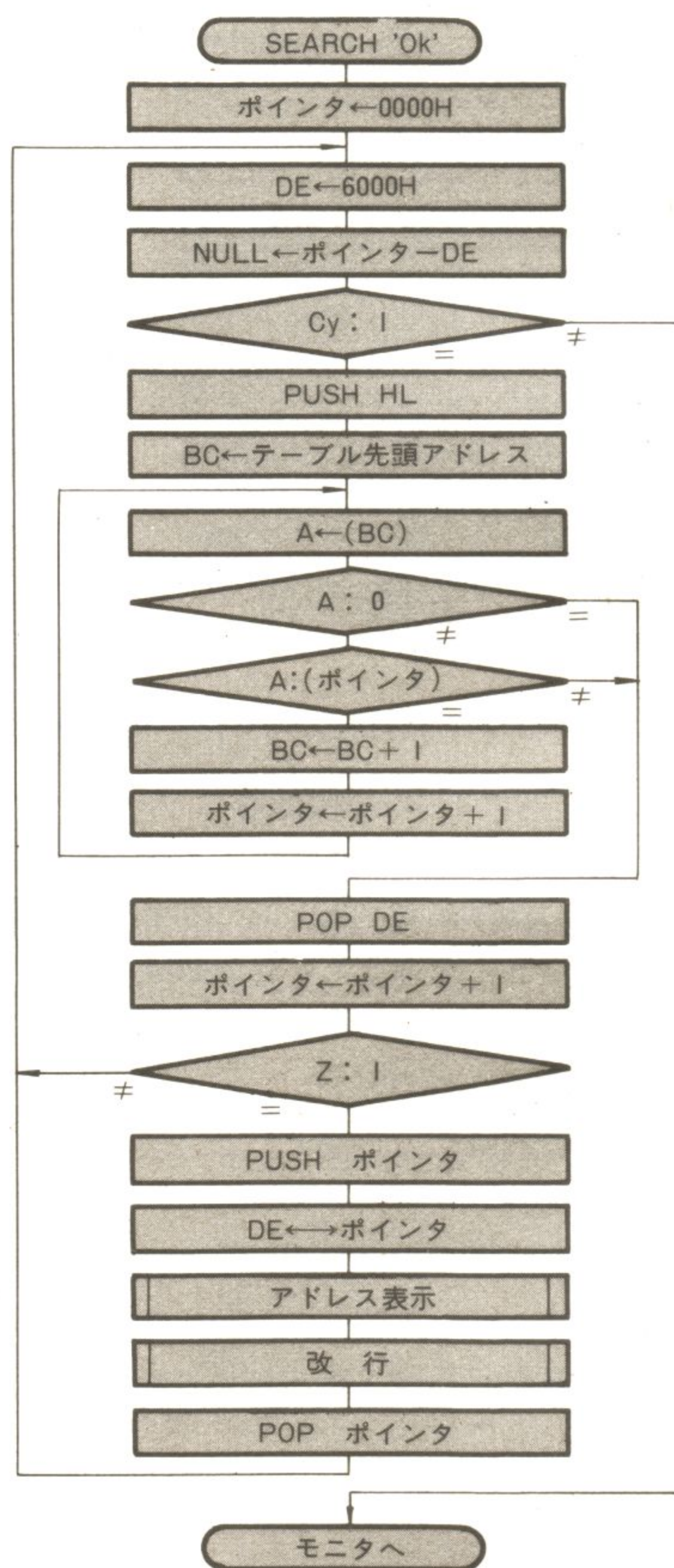
「せっかく第81図をつくったのに！」と残念に思いながらも、第81図を入力、実行したところ、当然のことですが3B60を表示し終わりました。

第83図は、ローテート命令を使った乗算サブルーチンです。HレジスタとLレジスタを掛け合わせた積をHLレジスタに求めます。Z80には16ビットの演算命令があるので

32ビットの乗算ルーチンも簡単にできると思います。

少なくとも、乗数を被乗数回加えるというアルゴリズムよりは速いと思いますが、いかがなものでしょうか？

《第82図》 解答例3のフローチャート



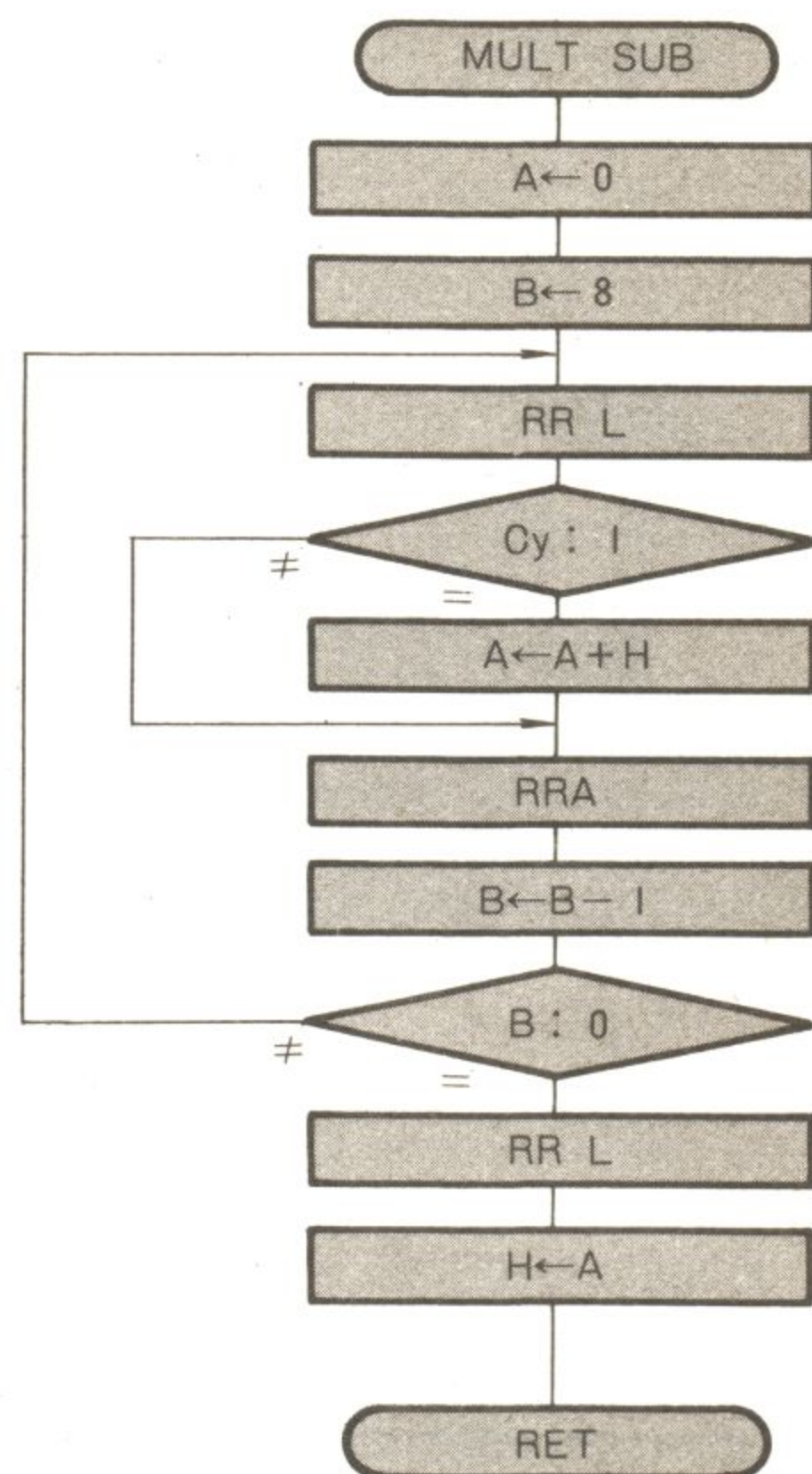
《第83図》乗算サブルーチン

《第84図》乗算サブルーチンのフローチャート

```

; *****
;
; 16 BIT MULTIPLY SUBROUTINE
;
; *****
;
; ORG 0E000H
;
E000 AF      XOR A
E001 0608    LD B, 8
;
E003 CB1D    LOOP: RR L
E005 3001    JR NC, NEXT
E007 84      ADD A, H
;
E008 1F      NEXT: RRA
E009 10FB    DJNZ LOOP
E00B CB1D    RR L
E00D 67      LD H, A
E00E C9      RET

```



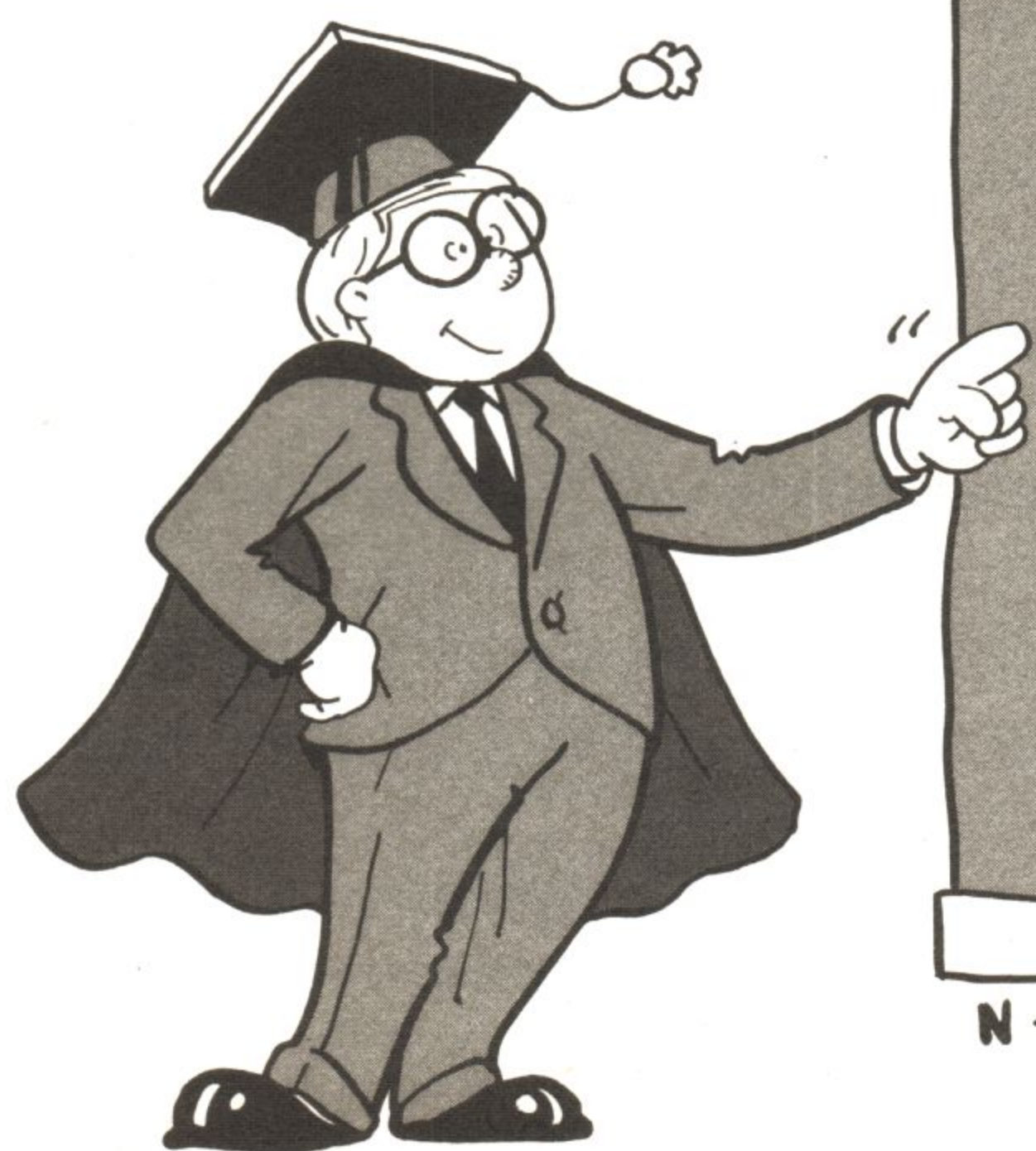
まとめ

N-BASICの中身も少しずつ公開され、マシン語でのプログラミングも比較的手軽になってきました。

本章のプログラム例でも、いくつかのROM内サブルーチンを利用する事によって、プログラム自体が簡潔にまとめられており、N-BASIC

等システム言語解析の重要性を感じてしまいます。

マシン語を使いこなせるようになったら、N-BASICなどを解析してみるのもおもしろいかも知れません。



52 EDH ----- 文字列の出力

5C 66H ----- マシン語モニタのホットスタート

5E COH ----- 16進4桁表示 (HLレジスタ対のデータを表示)

5E D3H ----- レジスタペアの比較 (HLレジスタ対とDEレジスタ対のデータ比較)

5F CAH ----- CRコードとLFコードの表示 (改行する)

N-BASICのシステムサブルーチン講座
(詳しくは付録 D, E 参照)

CPUコントロール命令

本章では残り少なくなった命令のなかから、CPUコントロール命令およびアキュムレータ操作命令を紹介します。

CPUコントロール命令

CPUコントロール命令は、CPUを直接コントロールするために用意された命令群で、CPUの停止や割り込みのコントロールを行うために7種類の命令が用意されています。

ただし、これらの7種の命令の中には一般にはほとんど使用されない命令もありますので、次にあげる2個の命令だけを理解しておいてください。

NOP

HALT

◎NOP

NOP命令は、

NO OPERATION

の言葉どおり、オペレーションを行わないための命令で、実行してもZ80は何も行わず、実行時間をわずかに消費するのみです。

実際には、プログラムのデバッグを行ってバイト数が少なくなりメモリが余ってしまった場合の穴埋めなどに利用します。

ハンド・アセンブルによってマシン語のプログラムを開発する場合には、デバッグ時にどうして

も途中で2～3バイト付け加える必要が生じる事があります。この様な時の事を考えて、プログラムの要所要所、特に各サブルーチンの最後には3バイト程度のNOP命令を入れておきましょう。3バイト空いていれば、サブルーチン・コール命令を使用する事もできますからほとんどの場合に対応することができます。

◎HALT

この命令は、本書のはじめの頃にプログラムの最後に置いて実行を停止するために利用していたものなので覚えておられる皆さんも多いでしょう。

HALT命令を実行したZ80は、HALT状態に入りHALT信号を送出します。この時CPUはNOP命令を実行し続けますが、このようにHALT状態の間NOP命令を実行する目的はPC-8001でもメインメモリとして使用しているD-RAM（ダイナミックラム）に不可欠なメモリのリフレッシュを行うためです。

HALT状態から抜け出すためには、PC-8001の背面にあるリセット・スイッチを押してRESET信号を発生させCPUをイニシャライズするか、割り込み信号によって割り込みを発生するしかありません。

なお、ニーモニックの読み方にもBASICのコマンドの読み方などの様にいろいろありますが、HALT命令は、

HALT（ホールド）

と読みます。私もつい最近までローマ字読みで、
HALT (ハルト)
 と読んでいました。皆さん、くれぐれも御注意を！

◎DI

Disable Interrupt の事で、マスク可能な割り込みを禁止します。

具体的には、インタラプト・イネーブル・フリップフロップ、**IFF1**および**IFF2**がリセットされます。

◎EI

Enable Interrupt の事で、マスク割り込み要求**INT**による割り込みを可能にする、**DI**命令と正反対の命令です。

具体的には、インタラプト・イネーブル・フリップフロップがセットされますが、当然、割り込みの実行中には他の割り込みは禁止されます。

◎IM 0

割り込みモードを0に設定します。モード0では8080等のCPUと同じく全ての割り込みを受け付けます。

◎IM 1

割り込みモードを1に設定します。モード1では、プログラム・カウンタ(PC)をスタックに退避した後0038H番地へのジャンプを行います。

◎IM 2

割り込みモードを2に設定します。モード2では、プログラム・カウンタ(PC)をスタックに退避したあと、インタラプト・ベクタ・レジスタ(I)で上位8ビットを指定し、割り込みをかけたデバイスが割り込みアクノリッジのサイクルでデータ・バス上に乗せた8ビットを下位とした、アドレスから2バイトに渡って書かれているデータをコール先アドレスとして、サブルーチン・コールを行います。

以上により、Z80では割り込み処理ルーチンのスタートアドレス最大128個を集めたテーブルを、メモリ上の任意のページにセットしておくこ

《第85図》ニーモック↔機械語対応表

CPUコントロール

アキュムレータ操作

| | |
|------|----------|
| NOP | 00 |
| HALT | 76 |
| DI | F3 |
| EI | FB |
| IM 0 | ED 46 |
| IM 1 | ED 56 |
| IM 2 | ED 5E |

| | |
|-----|----------|
| DAA | 27 |
| CPL | 2F |
| NEG | ED 44 |
| CCF | 3F |
| SCF | 37 |

とができます。

N-BASICでは、このモード2のためのテーブルを、RAM領域の一番先頭に置いてあるため、16KバイトRAMのPC-8001では、C000H番地からがテーブルとなります。

アキュムレータ操作命令

アキュムレータ操作命令は、Aレジスタやキャリーフラグ(CY)の操作を行うための命令群で5種類の命令が用意されています。

◎DAA

Z80や8080の加算および減算命令は、当然の事ながら2進数(16進数)の演算を行うために用意されていますから、BCDコードを直接使うためには注意が必要です。

例えば、BCDコードの15Hと14Hを加えた場合には、29Hとなり、一見問題が無い様に思えるのですが、15Hと15HをZ80の加算命令によって普通に加えると1AHが和として求められてしまいます。本当ならば20Hを求めたいのですが、この1AHから20Hへの補正を自動的に行うのが、

Decimal Adjust Accumulator

DAA命令です。

使い方は非常に簡単で、加算または減算命令によってBCDコードの演算を行った直後に必ずDAA命令を実行すれば良いのです。

◎CPL

CPL命令は、Aレジスタのもつ8個のビットの内容を全て反転するもので、つまりAレジスタのNOT (\bar{A}) を求める事ができる命令です。

たとえば、Aレジスタに

1 0 1 1 1 1 0 1

が収納されている時に、この命令を実行すると、

0 1 0 0 0 0 1 0

に変化します。

◎NEG

Aレジスタの内容が、符号付きの8ビットとして、-128~127の数値を表現している時に、その符号を反転するために用いる命令です。

具体的には、Aレジスタの各ビットを反転して1を加えますが、式に直すと

$$A \leftarrow \bar{A} + 1$$

となります。

◎CCF

この命令は、キャリーフラグ(CY)を反転するための命令です。

一般にキャリーフラグ(CY)を0にリセットするには、何らかの論理演算命令を実行すれば良く、レジスタの内容等に影響を与えない

AND A

などを行うのが一般的ですが、正式には次に説明する

SCF

によって、キャリーフラグを1にセットしておき、

CCF

で反転する事によって、0にリセットします。

BCDコード加減プログラム

DAA命令の使用例として、4バイトのBCDコードの間で加算および減算を行うプログラム例(第86図)を示します。4バイトのBCDコードでは、8桁の10進数を扱う事ができますから、正数であれば電卓並みの桁数で加減算ができるわけです。

プログラム自体は非常に簡単な構成なのでニーマニックを追って行くだけですぐに理解できる事と思いますから、プログラムの走らせ方だけを説明しておきましょう。

プログラムは、D000H番地から走らせる事によって最初に加算を、次に減算を行います。

和はE008H番地から4バイトに渡って、差はE00CH番地から4バイトに渡って、それぞれ収納され、E000H~E003H番地にわたるBCDコードが被加算でも被減算でもあり、E004H~E007H番地にわたるBCDコードが加数としても減数としても扱われます。

E000H~E007H番地に与える入力パラメータは、N-BASICのPOKEやマシン語モニタのSコマンドによってメモリ上にセットし、演算結果はマシン語モニタのDコマンドによって調べるのが一般的でしょう。

前章までのプログラム例と異なり結果をBCDコードで得る事ができますので、人間の立場でも非常にわかり易いと思います。

プログラム中で注意が必要なのは、SUB命令のオペランドが一つしかいない点で、ハンド・アセンブルの場合には良いのですが、アセンブラに頼る場合は不必要なオペランドを書くとエラーになります。

《第86図》BCDコード・テストプログラム (DAA命令の使用例)

```

; *****
;
;   BCD CODE TESTING PROGRAM
;
; *****
;
;   INPUTS   : (E003H)(E002H)(E001H)(E000H) <-BCD CODE 1
;               : (E007H)(E006H)(E005H)(E004H) <-BCD CODE 2
;
;   OUTPUTS  : (E00BH)(E00AH)(E009H)(E008H) <- SUM
;               : (E00FH)(E00EH)(E00DH)(E00CH) <- DIFFERENCE
;
;   ORG 0D000H
;
; *****
;
;               ADDITION
;
; *****
;
D000 DD2100E0      LD IX,0E000H
;
D004 DD7E00      LD A,(IX)
D007 DD8604      ADD A,(IX+4)
D00A 27          DAA
D00B DD7708      LD (IX+8),A
;
D00E DD7E01      LD A,(IX+1)
D011 DD8E05      ADC A,(IX+5)
D014 27          DAA
D015 DD7709      LD (IX+9),A
;
D018 DD7E02      LD A,(IX+2)
D01B DD8E06      ADC A,(IX+6)
D01E 27          DAA
D01F DD770A      LD (IX+10),A
;
D022 DD7E03      LD A,(IX+3)
D025 DD8E07      ADC A,(IX+7)
D028 27          DAA
D029 DD770B      LD (IX+11),A
;
; *****
;
;               SUBTRACTION
;
; *****
;
D02C DD7E00      LD A,(IX)
D02F DD9604      SUB (IX+4)
D032 27          DAA
D033 DD770C      LD (IX+12),A
;
D036 DD7E01      LD A,(IX+1)
D039 DD9E05      SBC A,(IX+5)
D03C 27          DAA
D03D DD770D      LD (IX+13),A
;
D040 DD7E02      LD A,(IX+2)
D043 DD9E06      SBC A,(IX+6)
D046 27          DAA
D047 DD770E      LD (IX+14),A
;
D04A DD7E03      LD A,(IX+3)
D04D DD9E07      SBC A,(IX+7)
D050 27          DAA
D051 DD770F      LD (IX+15),A
;
D054 C3665C      JP 5C66H

```


入出力命令

本章では、N-BASICの

INP

OUT

に値するZ80の入出力命令を紹介します。

これらの命令は、普段はあまり使われる事のない命令ですが、使用される場合には非常に重要でかつ注意を要する事が多いものです。

基本的には、CPU (Z80) に接続されている各種の周辺機器、たとえばキーボード、スピーカ、CRT、プリンタ、ディスク・ユニット等をコントロールするために使用される命令です。

入力命令

◎一般的な入力命令

マシン・コードの2バイト目で、入力を行うポートを指定する命令で、入力したデータはAレジスタに入ります。

たとえば、入力ポート21H番地からの入力データAレジスタに入力したい場合には、

```
IN    A, (21H)
```

を行う事になり、マシン・コードは、

```
DB 21
```

の2バイトになります。

◎Cレジスタによる入力ポート指定

一般的な入力命令では、マシン・コードの2バイト目で入力ポートのアドレスを指定しましたが、この方法では演算結果や条件によってポートのアドレスを変更したい場合に不便です。

そこで、Z80ではCレジスタの値を入力ポートのアドレスとして、ポートからの入力を行う命令が拡張されました。

ニーモニックは、Aレジスタに入力する、

```
IN    A, (C)
```

をはじめとして、

```
IN    B, (C)
```

```
IN    C, (C)
```

```
IN    D, (C)
```

```
IN    E, (C)
```

```
IN    H, (C)
```

```
IN    L, (C)
```

の計7種が用意されています。

これらの命令の必要性を確認するために、プログラム例を示しますが、このプログラム例はPC-8001上で走らせても意味は有りませんので御注意ください。

EXAMPLE-1からEXAMPLE-3までの3本は、全て同じ機能を持たせたプログラムです。

具体的には、入力ポート10H番地からのデータ入力を行い、入力したデータで指定する入力ポ

《EXAMPLE—1》

```

; *****
;
;      SAMPLE 1
;
; *****
;
;      ORG    0C100H
;
C100 DB10      IN    A, (10H)
C102 3206C 1    LD    (INPORT+1), A
;
C105 DB00      INPORT:IN    A, (00H)
;
C107 C9        RET

```

ートから再度入力を行うサブルーチンとなっています。

8080では、EXAMPLE—1の様にサブルーチン自身を書き換えるのが一般的ですが、このような手法はプログラムの見易さや、バグの発生率から考えてあまり勧められる方法ではありません。また、プログラムをROMに焼いて使用する場合などは、RAM上にジャンプ・テーブル等を置く必要があり、さらに複雑になってしまいます。かと言って、自分自身を書き換える事をさけて普通に組めば、EXAMPLE—2のような長大なプログラムになり、とても実用にはなりません。

そこで、EXAMPLE—3ですが、Cレジスタによる入力アドレス指定を利用しているため、非常に簡潔なものとなっています。

◎INI

INIは、

Input and Increment

の意味で、Cレジスタの内容で指定する入力ポートからの入力データを、HLレジスタ対で指定するアドレスに格納し、さらにHLレジスタ対をインクリメント、Bレジスタをデクリメントします。命令の実行結果Bレジスタが0になれば、ゼロ・フラグ(Z)を1にセットします。

この、

INI

と同様の機能を、今までに紹介した命令のみで表わすと、次のようになります。

```

PUSH AF
IN    A, (C)
LD    (HL), A
POP   AF

```

《EXAMPLE—2》

```

; *****
;
;      SAMPLE 2
;
; *****
;
;      ORG    0C100H
;
C100 DB10      IN    A,(10H)
C102 3C        INC    A
;
C103 3D        DEC    A
C104 2003      JR     NZ,NEXT1
C106 DB00      IN     A,(0)
C108 C9        RET
;
C109 3D        NEXT1:DEC A
C10A 2003      JR     NZ,NEXT2
C10C DB01      IN     A,(1)
C10E C9        RET
;
C10F 3D        NEXT2:DEC A
C110 2003      JR     NZ,NEXT3
C112 DB02      IN     A,(2)
C114 D9        RET
;
C115 3D        NEXT3:DEC A
C116 2003      JR     NZ,NEXT4
C11B DB03      IN     A,(3)
C11A C9        RET
;
C11B 3D        NEXT4:DEC A
C11C 2003      JR     NZ,NEXT5
C11E DB04      IN     A,(4)
C120 C9        RET
;
C121 3D        NEXT5:DEC A
C122 2003      JR     NZ,NEXT6
C124 DB05      IN     A,(5)
C126 C9        RET
;
C127 3D        NEXT6:DEC A
C128 2003      JR     NZ,NEXT7
C12A DB06      IN     A,(6)
C12C C9        RET
;
C12D 3D        NEXT7:DEC A
C12E 2003      JR     NZ,NEXT8
C130 DB07      IN     A,(7)
C132 C9        RET
;
C133          NEXT8:
;
;      (以下、同様に
;      続く)
;

```

《EXAMPLE—3》

```

; *****
;
;      SAMPLE 3
;
; *****
;
;      ORG    0C100H
;
C100 DB10      IN    A, (10H)
C102 4F        LD    C,A
;
C103 ED78      IN    A, (C)
;
C105 C9        RET

```



```
INC    HL
DEC    B
```

◎ INIR

INIRは、

Input, Increment and Repeat

の意味で、Bレジスタの内容が0になるまで

```
INI
```

をくり返す命令です。すなわち、Cレジスタで指定する入力ポートから、256バイト以内の入力データ・ブロックを、1命令でメモリ上に転送する事ができます。

この、

```
INIR
```

と同等の機能を他の命令で置き換えると、

```
PUSH AF
LBL: IN    A, (C)
LD        (HL), A
INC       HL
DJNZ     LBL
POP       AF
```

または、

```
LBL: INI
JR        NZ, LBL
```

などとなります。

◎ IND

INDは、

Input and Decrement

の意味で、ほぼ

```
INI
```

と同じですが、HLレジスタ対をインクリメントせずにデクリメントします。すなわち、Cレジスタの内容で指定する入力ポートからの入力データを、HLレジスタ対で指定するアドレスに格納し、さらにHLレジスタ対とBレジスタをデクリメントする事になります。命令の実行結果Bレジスタの内容が0になれば、ゼロ・フラグ(Z)が1にセットされます。

この、

```
IND
```

と同様の機能を今までに紹介した命令のみで表わすと、

```
PUSH AF
IN      A, (C)
LD      (HL), A
POP     AF
DEC     HL
DEC     B
```

または、

```
INI
DEC     HL
DEC     HL
```

などとなります。

◎ INDR

INDRは、

Input, Decrement and Repeat

の意味で、Bレジスタの内容が0になるまで、

```
IND
```

をくり返す命令です。すなわち、Cレジスタで指定する入力ポートから、256バイト以内の入力データ・ブロックを、1命令でメモリ上に転送する事ができます。

この、

```
INDR
```

と同等の機能を他の命令で置き換えると、

```
PUSH AF
LBL: IN    A, (C)
LD        (HL), A
DEC       HL
DJNZ     LBL
POP       AF
```

または、

```
LBL: IND
JR        NZ, LBL
```

などとなります。

出力命令

◎一般的な出力命令

マシン・コードの2バイト目で、出力を行うポートを指定する命令で、Aレジスタのデータが出力ポートに出力されます。

たとえば、PC-8001では出力ポート51H番地に21Hを出力する事によって、CRTスクリーンを反転（リバーズ）できる事が良く知られていますが、これをニーモニックにすると、

```
LD      A, 21H
OUT     (51H), A
```

となります。

ちなみに、反転したCRTスクリーンをノーマルな状態にもどすためには、普通、出力ポート51H番地にデータ20Hを出力しますが、これには、

```
LD      A, 20H
OUT     (51H), A
```

を行います。

◎Cレジスタによる出力ポート指定

一般的な出力命令では、マシン・コードの2バイト目で出力ポートのアドレスを指定し、Aレジスタのデータを出力しましたが、この方法では、演算結果や条件によってポートのアドレスを変更したい場合に不便です。

そこで、Z80では入力命令同様にCレジスタの値を出力ポートのアドレスとして、ポートへの出力を行う命令が拡張されました。

ニーモニックは、Aレジスタのデータを出力する、

```
OUT     (C), A
```

をはじめとして、

```
OUT     (C), B
OUT     (C), C
OUT     (C), D
OUT     (C), E
OUT     (C), H
OUT     (C), L
```

の計7種が用意されています。

◎OUTI

OUTIは、

Output and Increment

の意味で、HLレジスタで指定するアドレス上のデータをCレジスタで指定する出力ポートに出力し、さらにHLレジスタ対をインクリメント、Bレジスタをデクリメントします。命令の実行結果Bレジスタが0になれば、ゼロ・フラグ（Z）が1にセットされます。

この、

OUTI

と同様の機能を今までに紹介した命令のみで表わすと、次のようになります。

```
PUSH AF
LD      A, (HL)
OUT     (C), A
POP     AF
INC     HL
DEC     B
```

◎OTIR

OTIRは、

Output, Increment and Repeat

の意味で、Bレジスタの内容が0になるまで、

OUTI

をくり返す命令です。すなわち、メモリ上に置かれている256バイト以内のデータ・ブロックを、Cレジスタで指定する出力ポートに1命令で出力する事ができます。

この、

OTIR

と同等の機能を他の命令に置き換えると、

```
PUSH AF
LBL: LD  A, (HL)
      OUT (C), A
      INC HL
      DJNZ LBL
      POP AF
```

または、

```
LBL: OUTI
      JR  NZ, LBL
```


などとなります。

◎OUTD

OUTDは、
Output and Decrement

の意味で、ほぼ

OUTI

と同じですが、HLレジスタ対をインクリメントせずにディクリメントします。すなわち、HLレジスタ対で指定するアドレス上のデータをCレジスタで指定する出力ポートに出力し、さらにHLレジスタ対とBレジスタをディクリメントする事になります。命令の実行結果Bレジスタの内容が0になれば、ゼロ・フラグ(Z)が1にセットされます。

この、

OUTD

と同様の機能を、今までに紹介した命令のみで表わすと、

```
PUSH AF
LD A, (HL)
OUT (C), A
POP AF
DEC HL
DEC B
```

または、

```
OUTI
DEC HL
DEC HL
```

などとなります。

◎OTDR

OTDRは、

Output, Decrement and Repeat

の意味で、Bレジスタの内容が0になるまで、

OUTD

をくり返す命令です。すなわち、メモリ上に置かれている256バイト以内のデータ・ブロックを、Cレジスタで指定する出力ポートに1命令で出力する事ができます。

この、

OTDR

と同等の機能を他の命令で置き換えると、

```
PUSH AF
LBL: LD A, (HL)
OUT (C), A
DEC HL
DJNZ LBL
POP AF
```

または、

```
LBL: OUTD
JR NZ, LBL
```

などとなります。

キーボード・ スキャンングの実際

一般に高速性を要するゲーム等で、リアルタイムなキー入力を行う場合には、1文字入力等のシステム・サブルーチンでは役不足です。

そこで、本章で紹介したポート入力の命令など

《第87図》ニーモニック↔マシン語対照表

| 入力 | | 出力 | |
|-----------|------------------|------------|------------------|
| IN A, n | DB _n | OUT n, A | D3 _n |
| IN A, (C) | ED ₇₈ | OUT (C), A | ED ₇₉ |
| IN B, (C) | ED ₄₀ | OUT (C), B | ED ₄₁ |
| IN C, (C) | ED ₄₈ | OUT (C), C | ED ₄₉ |
| IN D, (C) | ED ₅₀ | OUT (C), D | ED ₅₁ |
| IN E, (C) | ED ₅₈ | OUT (C), E | ED ₅₉ |
| IN H, (C) | ED ₆₀ | OUT (C), H | ED ₆₁ |
| IN L, (C) | ED ₆₈ | OUT (C), L | ED ₆₉ |
| INI | ED _{A2} | OUTI | ED _{A3} |
| INIR | ED _{B2} | OTIR | ED _{B3} |
| IND | ED _{A4} | OUTD | ED _{AB} |
| INDR | ED _{BA} | OTDR | ED _{BB} |

を利用して、キーボードの接続されているキー・ポートから直接データを入力して、特定のキーが押されているか否かを調べるのが普通です。

第88図に、PC-8001のキー入力ポートを示します。キーボードのそれぞれのキーは、このようなマトリクスで構成されており、おのこの列がデータ・バスに直接接続されていますので、入力ポートからの入力によって、簡単に押されたキーを判別する事ができます。

実際には、入力命令によってインプットアドレスの値を入力し、キーが押されていれば、その列のデータ・バスに対応したビットが0として入力されます。

実例をいくつかあげますと、たとえば、**[K]**のキ

ーが押されている場合に、

```
IN      A, (03H)
```

を行えば、Aレジスタに2進数の、

```
1111 0111
```

が入力され、**[O]**と**[K]**が同時に押されている場合に、

```
IN      A, (03H)
```

を行えば、

```
0111 0111
```

が入力される事になります。

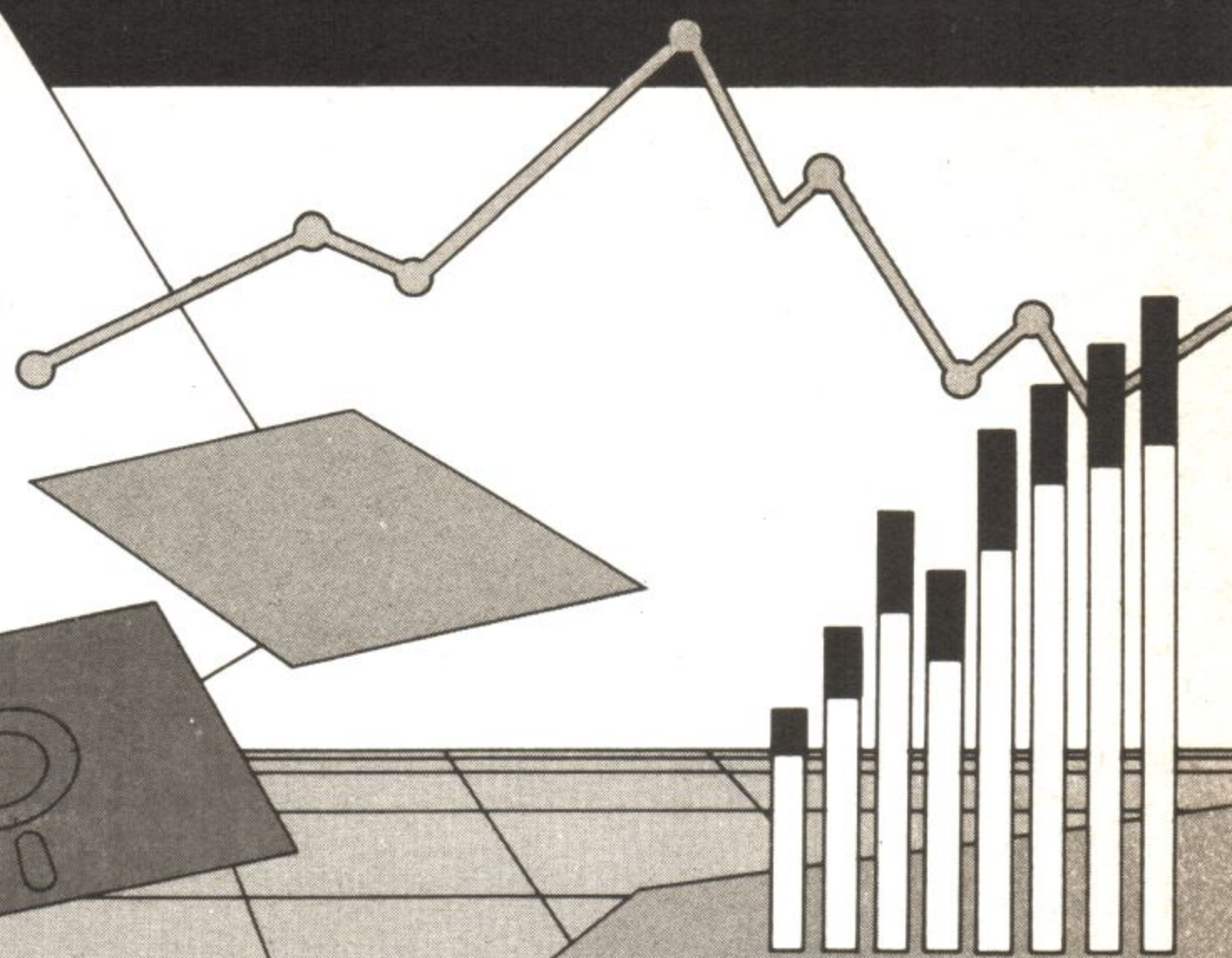
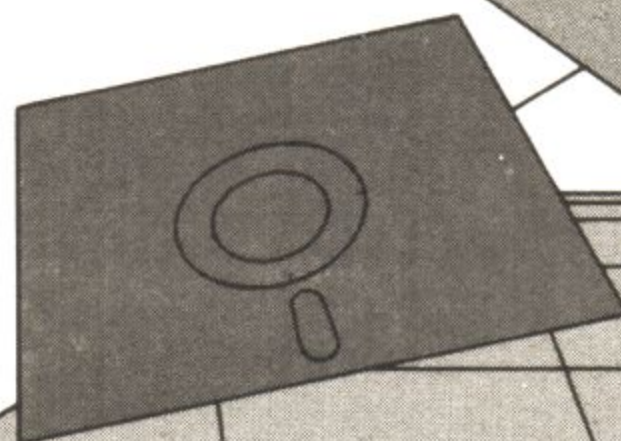
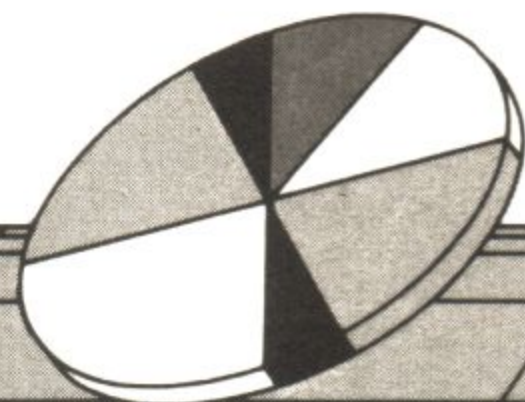
第87図として入出力命令のニーモニック→マシン語対照表を示しておきます。

《第88図》PC-8001のキーボード・マトリクス

| | | (データ・バス) | | | | | | | |
|----|--|-------------|--------|-------------|-------------|-------------|-------------|-------------|--------|
| | | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
| 00 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 01 | | 8 | 9 | * | + | = | , | . | RETURN |
| 02 | | @ " | A チ | B コ | C ソ | D シ | E イ | F ハ | G キ |
| 03 | | H ク | I ニ | J マ | K ノ | L リ | M モ | N ミ | O ラ |
| 04 | | P セ | Q タ | R ス | S ト | T カ | U ナ | V ヒ | W テ |
| 05 | | X サ | Y ン | Z ツ | [「 | ¥ — |] 」 | ^ へ | _ = |
| 06 | | 0 ワ | 1 ヌ | 2 フ | 3 ア | 4 ウ | 5 エ | 6 オ | 7 ヤ |
| 07 | | 8 ユ | 9 ヨ | : * ケ | : + レ | < , ネ | > . ル | ? / メ | - ロ |
| 08 | | HOME CLR | ↓ ↑ | ← → | INS DEL | GRAPH | カナ | SHIFT | CTRL |
| 09 | | STOP | f・1 | f・2 | f・3 | f・4 | f・5 | SPACE | ESC |
| | | (インプットアドレス) | | | | | | | |

| | D 0 | D 1 | D 2 | D 3 | D 4 | D 5 | D 6 | D 7 |
|----|-----|-----|-----|-----|-------|-----|-----|-----|
| 00 | | | | | | | | |
| 01 | | | | | | | | |
| 02 | | | | | | | | |
| 03 | | | | | | | | |
| 04 | | | | | | | | |
| 05 | | | | | | | | |
| 06 | 秒 | | | | | 年 | 月 | 日 |
| 07 | 時 | 分 | | | | | | |
| 08 | | | | | GRAPH | | | |
| 09 | | | | | | | | |

マシン語Q&A



Z80インストラクションの解説も、ブロック転送命令／ブロック・サーチ命令／ビット操作命令を残すだけになりました。そこで本章では、今までのまとめの意味も含めて、月刊マイコン編集部に寄せられたマシン語の質問にお答えしていきたいと思います。

Question 1

最近、BASICにもソースプログラムとオブジェクトコードがあると聞きました。『ソース』とか『オブジェクト』とか言うのはアセンブラ(マシン語)にのみ適用すると思っていたのですが、高級言語にもその様な考え方があるのでしょうか？

また、アセンブラで言う『ソースプログラム』とは単に逆アセンブラをしただけで得られるものなのでしょうか？ SE(システム・エンジニア)をやっている私の友人は『ソースリスト』の重要性をうたえてくれるのですが私には良くわかりません。

初心者にもわかるように解説をお願いします。

(東京都 納谷佳子さん)

Answer 1

最近、マイコンショップなどで、『ソースリスト』や『ソースプログラム』という言葉が流行しているらしく、私のところにも、

『ソースとは何の事ですか？』

というたぐいの質問が、多数寄せられています。

御質問の中でも述べられているように、『ソースプログラム』はアセンブラだけの考え方ではありません。

皆さん良く御存知のように、コンピュータで動く言語には、N-BASICの様なインタプリタの他に、プログラム全体を一度マシン語(オブジェクトコード)に変換してから実行するコンパイラ型の言語が多数あります。

この、オブジェクトコードに変換する前のプログラムを『ソースプログラム』と言うのですから、コンパイラ型の言語であれば、BASICであれ、FORTRANであれ、PASCALであれ全て『ソースプログラム』と『オブジェクトコード』の二つが存在します。以上の定義によるとコンパイラ型で無いアセンブラ『ソースプログラム』などが、存在するのはおかしいと思われるかもしれませんが正確にはアセンブラもコンパイラの一種なのです。

しかし、アセンブラが他のコンパイラと異なる点は『ソースプログラム(ニーモニック)』と『オ

プロジェクトコード(マシン語)』が1対1に対応している事で、そこがアセンブラの最大の利点であり欠点でもあるのです。

それでは、アセンブラにおける『ソースプログラム』の考え方について少し深く考えてみましょう。

『ソース』のスペリングは、『SOURCE』で、その意味を英和辞典などで調べてみると、普通、

- ①源, 源泉, 水源。
- ②もと, 起り, 原因, 根元。
- ③(情報などの)出所, よりどころ, 典拠。

と掲載されています。

つまり、この『ソースリスト』や『ソースプログラム』を直訳すると、『もとのリスト』とか『根元のプログラム』などと訳せるわけです。

普通、Z80などを用いるマシン語のプログラムを組む場合には、本書でも実践して来たように、一度ニーモニックでコーディングしてからアセンブルの作業によってマシンコードを割り当てて行きます。

ところが、マイコン関係の雑誌上に掲載されたり、テープやディスクによって供給されるソフトウェアは、ニーモニックの部分を省いて16進数のラ列と化したものが大部分です。この、アドレスと16進数のラ列であるマシンコードのみから成るリストを『ダンプリスト』と言います。

単にプログラムを実行するだけであれば、『ダンプリスト』のみで問題ないのですが、プログラムを読んだり(解析したり)、改良したりしようと思った場合には、『ダンプリスト』を追って行くわけにはいきません。ある程度慣れて来れば、マシン語とニーモニックの関係が覚えられますが、それでも長大なプログラムを『ダンプリスト』のみで追って行く事は不可能です。

そこで、このようなマシン語のプログラムを、マシン語とニーモニックの対照表を引くなり、逆アセンブラ(ディスアセンブラ)を利用するなりして、ニーモニック付きのリストに変換しますが、このようにして逆アセンブルされたものを、『逆アセンブルリスト』(俗称逆アセリスト)などと呼びます。

『逆アセンブルリスト』になれば、プログラムの流れを追う事が比較的簡単になります。しかし、普通に逆アセンブルしたリストには多くの問題が残されています。次に、それらの主な問題点をあげます。

《第89図》ダンプリスト

PC-8001内蔵マシン語モニタのDコマンド等によって得られます。この『ダンプリスト』のみで長いプログラムの流れを追うのは困難です。

| | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| C000 | 06 | 50 | 0E | 19 | CD | 3A | 09 | 21 | 00 | F3 | 06 | 19 | E5 | 11 | 25 | C0 |
| C010 | 1A | FE | 00 | 28 | 06 | 77 | 13 | 23 | C3 | 10 | C0 | E1 | 11 | 78 | 00 | 19 |
| C020 | 10 | EA | C3 | 66 | 5C | 2A | 2A | 2A | 20 | B9 | DE | AF | B6 | DD | 20 | CF |
| C030 | B2 | BA | DD | 20 | 2D | 2D | 2D | 20 | 50 | 43 | 2D | 38 | 30 | 30 | 31 | 20 |
| C040 | 4D | 41 | 43 | 48 | 49 | 4E | 45 | 20 | 43 | 4F | 44 | 45 | 20 | 2D | 2D | 2D |
| C050 | 20 | CF | BC | DD | BA | DE | 20 | BA | B3 | BB | DE | 20 | B7 | BF | CD | DD |
| C060 | 20 | 2D | 2D | 2D | 20 | 51 | 20 | 61 | 6E | 64 | 20 | 41 | 20 | BA | B0 | C5 |
| C070 | B0 | 20 | 2A | 2A | 2A | 00 | | | | | | | | | | |

《第90図》アスキーダンプ付きダンプリスト

『ダンプリスト』の右側にキャラクタによるダンプを加えたものです。メモリ上のメッセージなどをさがし出してデータエリアを分離するのに役立ちます。

| | | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------------------------|
| C000 | 06 | 50 | 0E | 19 | CD | 3A | 09 | 21 | 00 | F3 | 06 | 19 | E5 | 11 | 25 | C0 | . P . . . : . ! % . |
| C010 | 1A | FE | 00 | 28 | 06 | 77 | 13 | 23 | C3 | 10 | C0 | E1 | 11 | 78 | 00 | 19 | . . . (. W . # . . . X . . |
| C010 | 10 | EA | C3 | 66 | 5C | 2A | 2A | 2A | 20 | B9 | DE | AF | B6 | DD | 20 | CF | . . . f ¥ * * * ゲッカン マ |
| C030 | B2 | BA | DD | 20 | 2D | 2D | 2D | 20 | 50 | 43 | 2D | 38 | 30 | 30 | 31 | 20 | アイコン ——— PC-8001 |
| C040 | 4D | 41 | 43 | 48 | 49 | 4E | 45 | 20 | 43 | 4F | 44 | 45 | 20 | 2D | 2D | 2D | MACHINE CODE ——— |
| C050 | 20 | CF | BC | DD | BA | DE | 20 | BA | B3 | BB | DE | 20 | B7 | BF | CD | DD | マシンゴ コウザ キソヘン |
| C060 | 20 | 2D | 2D | 2D | 20 | 51 | 20 | 61 | 6E | 64 | 20 | 41 | 20 | BA | B0 | C5 | ——— Q and A コーナ |
| C070 | B0 | 20 | 2A | 2A | 2A | 00 | | | | | | | | | | | — * * * . |

- ①ラベルが付いていない。
- ②データエリアの分離が不十分。
- ③コメントが付いていない。

①のラベルが付いていない事は、プログラムの動きを知る上での障害となり、プログラムを読む（解析する）場合には適切なラベルを付ける事で能率が飛躍的に向上します。たとえば、『●』をCRTスクリーンに表示するプログラムで、

```
LD      A, '●'
CALL    0257H
```

と書いてあるのと、

```
LD      A, '●'
CALL    DSPCHR
```

と、0257H番地のディスプレイ・キャラクタ・サブルーチンに、『DSPCHR』のラベルを定義して参照するのとでは、ニュアンスが全く異なります。

もう一つ例をあげますと、

```
LD      HL, 1838H
CALL    52EDH
```

に適切なラベルを付けると、

```
LD      HL, TITLE
CALL    DSPMSG
```

などとなります。この例で、『TITLE』とラベルを付けた1838H番地から後にはN-BASICをコールド・スタートした時のタイトルメッセージ

```
NEC PC-8001 BASIC. Ver 1, 1
Copyright 1979 (C) by Microsoft
```

が格納されており、『DSPMSG』とラベルを付けた52EDH番地は文字列（メッセージ）を出力するためのサブルーチンとなっています。

②のデータを正確に分離する事の必要性につきましては、二つのリスト（第91図と第92図）後半のデータ部分を比較していただければ一目でわかるでしょう。この2本のプログラムは、CRTスクリーンを80文字×25桁モードに設定して、全体を、

```
*** ゲッカン マイコン ----- PC-8001
MACHINE CODE ----- マシンゴウザ キソヘン
----- Q and A コーナー ***
```

の文字で埋める全く同じプログラムですが、前者は普通に逆アセンブルしたものです。一応データエリアの分離は行っていますが、16進のキャラクタコードとなっているために、表示する文字列のイメージが頭に浮かんで来ません。

なお、ここで使っている

DB

と言う命令はアセンブラでのみ利用できるもので、オペランドで指定する16進数またはアポストロフィー『』で囲んだ文字列のキャラクタコードをオブジェクトのメモリ上にそのまま落とすものです。

アセンブラを使用してマシン語のプログラムを組んで行く場合には、必要に応じてコメント（注釈）を入れておきます。このコメントはBASICのREM（リマーク）と同じように、プログラムの実行時（アセンブル時）には全く無視されるもので、普通のアセンブラではセミコロン『;』の後、行末までがコメントとみなされます。

コメントを入れるか否か、入れるとすればどの程度入れるのかは、プログラマの主観によるのですが、一般的には、ルーチンの切れ目などの特に重要な部分や特殊技法などを使ってプログラムがわかりずらくなった部分には適切なコメントが必要です。

Question 2

プログラムを組む前にフローチャートを書くように心がけている者です。

ゼネラルフローを書く場合には、FORTRANやPASCAL, BASICなどと変わらないのですが、アセンブラのインストラクション・レベルでフローチャートを書く場合にはどのような記号を用いるのでしょうか？

ロード命令などは、通常の代入と同じ矢印『←』を使えば良いと思うのですが、PUSH/POP

《第92図》ソースリスト

アセンブラの『ソースプログラム』です。ラベルやコメントも入り、プログラマの意図が適確に伝わる非常に追い易いリストです。

```

; ****
;
;       メッセージ  25   カイ   ヒョウジ
;
; ****
093A      WIDTH: EQU  093AH      ;『WIDTH』ニ WIDTH ノ セッテイ サブ ルーチン チ テイギ
5C66      MONTOP: EQU 5C66H     ;『MONTOP』ニ マシゴ モニタ ノ セントウ バンチ チ テイ
          ORG   0C000H
;
C000      0650      LD   B, 80      ;
C002      0E19      LD   C, 25      ; CRT スクリーン チ 80×25 モード ニ セット
C004      CD3A09    CALL WIDTH      ;
;
C007      2100F3    LD   HL, 0F300H ; VRAM ノ セントウ バンチ チ HL ニ セット
C00A      0619      LD   B, 25      ; 25 カイ ノ ループ チ シテイ
;
C00C      E5        LINE:  PUSH HL      ; HL ノ アライ チ スタック ニ タイヒ
C00D      1125C0    LD   DE, DATA     ; メッセージ ノ セントウ バンチ チ DE ニ セット
;
C010      1A        LETTER: LD   A, (DE) ; 1 モジ ブン ノ データ チ モッテ クル
;
C011      FE00      CP   0            ; データ ガ 0 デ アレ バ
C013      2806      JR   Z, LINEND    ;   ゲンザイ ノ ライン チ シュウリョウ
;
C015      77        LD   (HL), A      ; 1 モジ ブン ノ データ 1 モジ チ VRAM ニ オクル
;
C016      13        INC  DE           ; メッセージ ノ アドレス シテイ チ 1 バイト ススメル
C017      23        INC  HL          ; VRAM ノ アドレス シテイ チ 1 バイト ススメル
C018      C310C0    JP   LETTER      ; ツギ ノ モジ チ ヒョウジ スル タメ ニ ジャンプ
;
C01B      E1        LINEND: POP  HL      ;
C01C      117800    LD   DE, 120      ; VRAM ノ アドレス シテイ チ ツギ ノ ギョウ ニ ススメル
C01F      19        ADD  HL, DE       ;
;
C020      10EA      DJNZ LINE         ; ラベル 『LINE』 ヘ ループ
C022      C3665C    JP   MONTOP      ; PC—8 0 0 1 マシゴ モニタ ヘ ジャンプ
;
; ****
;
;       メッセージ データ
;
; ****
C025      2A2A2A20  DATA:  DB  , ***   ゲツカン   マイコン   —— ,
C029      B9DEAFB6
C02D      DD20CFB2
C031      BADD202D
C035      2D2D20
C038      50432D38      DB  , PC—8 0 0 1  MACHINE CODE  —— ,
C03C      30303120
C040      4D414348
C044      494E4520
C048      434F4445
C04C      202D2D2D
C050      20
C051      CFBCDDBA      DB  , マシゴ コウザ   キソヘン   —— ,
C055      DE20BAB3
C059      BBDE20B7
C05D      BFCDDD20
C061      2D2D2D20
C065      5120616E      DB  , Q  and  A   コーナー   *** ,
C069      64204120
C06D      BAB0C5B0
C071      202A2A2A
;
C075      00           DB   0
;

```

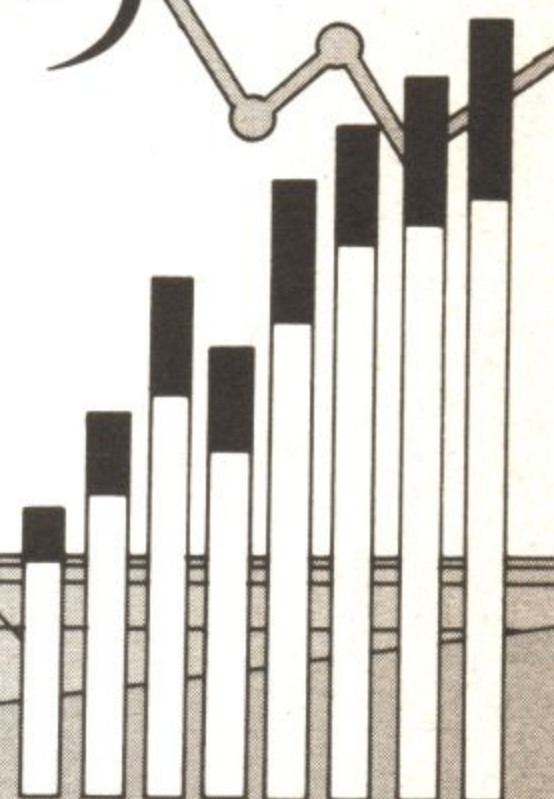
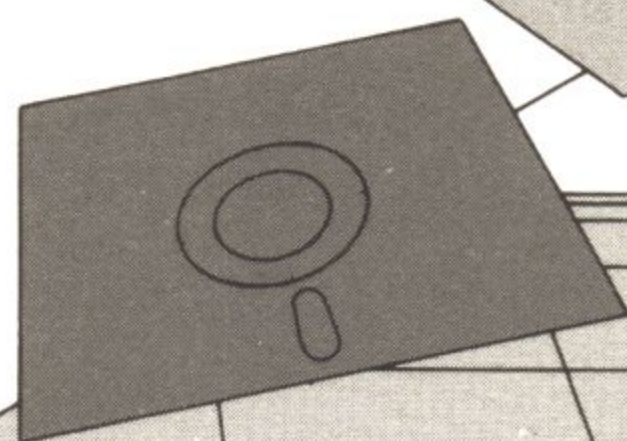
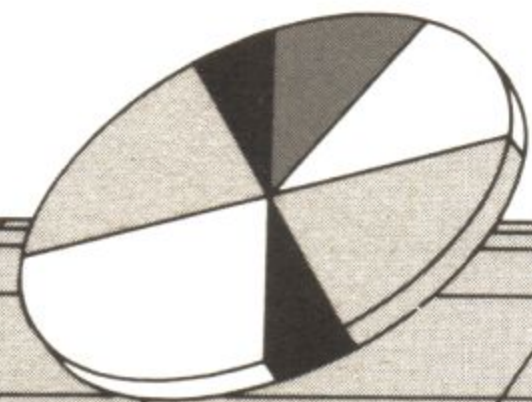

〈第94図〉 川村流ニーモニック↔フローチャート使用例 I

| ニーモニック | フローチャート | ニーモニック | フローチャート |
|---------------|------------------------------|------------|------------------------------|
| LD E, A | $E \leftarrow A$ | CPL | $A \leftarrow \bar{A}$ |
| LD A, '●' | $A \leftarrow \text{'●'}$ | NEG | $A \leftarrow -A$ |
| LD (E000H) HL | $(E000H) \leftarrow HL$ | CCF | $CY \leftarrow \bar{CY}$ |
| LD (HL), A | $(HL) \leftarrow A$ | SCF | $CY \leftarrow I$ |
| LD BC, 120 | $BC \leftarrow 120$ | SBC HL, DE | $HL \leftarrow HL - DE - CY$ |
| PUSH AF | PUSH AF | DEC IY | $IY \leftarrow IY - 1$ |
| POP DE | POP DE | RLCA | RLCA |
| LDIR | LDIR | RRD | RRD |
| CPD | CPD | JP 5C66H | 5C66H |
| ADD A, 10H | $A \leftarrow A + 10H$ | CALL 0257H | 0257H |
| ADC A, (HL) | $A \leftarrow A + (HL) + CY$ | IN A, (0) | IN A, (0) |
| AND A, C | $A \leftarrow A \wedge C$ | OUT (C), L | OUT (C), L |
| CP 12H | $NULL \leftarrow A - 12H$ | OUTD | OUTD |
| INC A | $A \leftarrow A + 1$ | RST 18H | RST 18H |
| EI | EI | SET 4, A | $A_4 \leftarrow 1$ |
| DAA | DAA | RET | RET |

〈第95図〉 川村流ニーモニック↔フローチャート使用例 II

| ニーモニック | フローチャート |
|--|---------|
| LOOP : CALL SUB 1 CP C JR NZ, LOOP RET | |
| LD C, 100 WAIT 2 : LD B, 100 WAIT 1 : CALL WAIT DEC B JR NZ, WAIT 1 DEC C JR NZ, WAIT 2 JP (HL) | |
| ADD A, A RET C | |

ブロック転送命令/ブロック・サーチ命令/ビット操作命令



ブロック転送命令

Z80に用意された、データ・ブロック転送命令はメモリ上に置かれている任意のバイト数のデータを、別のアドレスへ転送する際に利用される命令で、全て（4種）のブロック転送命令において3組のレジスタ対の使用目的（意味）が決まっています。

ブロック転送の命令において、HLレジスタ対は転送するデータの元のアドレスを、DEレジスタ対は転送先のアドレスを示し、BCレジスタ対はデータの量をカウントするためのバイト・カウンタとして使用されます。

◎LDI

LDIは、
Load and Increment
の意味です。

この命令を実行すると、HLレジスタ対で指定されたメモリの内容がCPUに読み込まれ、次にDEレジスタ対で指定するメモリに同じデータを転送します。データの転送後は、HLレジスタ対とDEレジスタ対の内容が自動的にインクリメントされ、バイト・カウンタであるBCレジスタ対はディクリメントされます。

同じような動作を他の命令に置き換えると、

PUSH AF

LD A, (HL)
LD (DE), A
POP AF
INC HL
INC DE
DEC BC

となりますが、ブロック転送命令ではフラグの変化が独特で、重要な意味を持っているので注意が必要です。

この命令は、一般的に利用されるサイン・フラグ(S)、ゼロ・フラグ(Z)およびキャリ・フラグ(CY)には全く影響を及ぼしませんが、BCレジスタ対をディクリメントした結果が0(ゼロ)になればパリティ/オーバーフロー・フラグ(P/V)も0にリセットされます。

したがって、ブロック転送したいデータのバイト数をBCレジスタ対に与えておけば、命令の実行後にP/Vフラグの値を調べる事によって命令の実行回数をチェックする事ができます。

◎LDIR

LDIRは、
Load, Increment and Repeat
の名称のとおり、先に紹介した

LDI
を、BCレジスタ対の内容が0(ゼロ)になってパリティ/オーバーフロー・フラグ(P/V)が0にリセットされるまでくり返す命令です。

他の命令で同じ機能を書き換えれば、

```
REPEAT:LDI
        JP PE, REPEAT
```

となります。

このように、

```
LDIR
```

を用いれば1ステップでデータのブロック転送を行う事ができます。例として、0000H番地から00FFH番地まで0100Hバイトのデータを、C000H番地からC0FFH番地に転送するルーチンを示します。

```
LD      HL, 0000H
LD      DE, C000H
LD      BC, 0100H
LDIR
```

◎LDD

LDDは、

Load and Decrement

の意味です。

```
LDI
```

と同じように、HLレジスタ対で指定するメモリの内容をDEレジスタ対で指定するメモリに転送しますが、転送後は、HLレジスタ対、DEレジスタ対およびBCレジスタ対の内容を全てディクリメントします。

同じような動作を他の命令に置き換えると、

```
PUSH AF
LD      A, (HL)
LD      (DE), A
POP     AF
DEC     HL
DEC     DE
DEC     BC
```

または、

```
LDI
DEC     HL
DEC     HL
DEC     DE
DEC     DE
```

のようになります。

しかし、パリティ/オーバーフロー・フラグ(P/V)

が重要な役割りを果たし、

```
LDI
```

と同じように、命令の実行結果BCレジスタ対が0(ゼロ)になった場合にP/Vフラグが0にリセットされます。

◎LDDR

LDDRは、

Load Decrement and Repeat

の意味で、

```
LDIR
```

と同じように1ステップでデータ・ブロック転送を行う事のできる命令です。具体的には、

```
LDD
```

をBCレジスタ対で指定する回数だけくり返すものですから、HLレジスタ対およびDEレジスタ対には、転送元と転送先の上位のアドレス(FFFFHに近いアドレス)を与えなくてははいけません。

たとえば、0000H番地から00FFH番地まで0100Hバイトのデータを、C000H番地からC0FFH番地に転送する場合は次のようになります。

```
LD      HL, 00FFH
LD      DE, C0FFH
LD      BC, 0100H
LDDR
```

ブロック・サーチ命令

メモリ・ブロック・サーチ命令は、Aレジスタの内容とメモリ上のデータとを比較するために利用される命令で、HLレジスタ対がデータのアドレスを指すポインタとして、BCレジスタ対は比較回数をカウントするバイト・カウンタとして使用されます。

Z80にはブロック・サーチのための命令が4種ほど用意されていますが、一般的にはあまり活用されていないようです。データ・ブロックの中から一定のデータをさがし出す使い方以外にも、最大値や最小値などを求める様な応用も出来ますので、サーチ以外にもデータのソーティング等を行う時に使ってみてください

◎CPI

CPIは、

Compare and Increment

の意味で、Aレジスタの内容とHLレジスタ対で指定するメモリの内容を比較して、比較結果をサインフラグ(S)およびゼロ・フラグ(Z)に表します。比較した後はHLレジスタ対をインクリメント、BCレジスタ対をデクリメントし、もしBCレジスタ対の内容が0(ゼロ)になればパリティ/オーバーフロー・フラグ(P/V)を0にリセットします。

他の命令に置き換えれば、

CP A, (HL)

INC HL

DEC BC

などとなりますが、フラグの変化は独自のものです。特に、ブロック・サーチ命令ではキャリ・フラグ(CY)に全く影響を及ぼしませんので注意が必要です。

◎CPIR

CPIRは、

Compare, Increment and Repeat

の名称のとおり、先に紹介した。

〈第96図〉CPIR命令の使用例(内蔵マシン語モニタの一部)

| | | | | | |
|------|----------|---------|-------------|-------------------------------|-----------------------------------|
| 5C2C | F3 | MONENT: | DI | | |
| 5C2D | 3AFF7F | LD | A,(EMONSW) | | |
| 5C30 | FE55 | CP | 55H | | ;CHECK EXTENDED MONITOR SW |
| 5C32 | CAFC7F | JP | Z, EMONSW-3 | | |
| 5C35 | 2234FF | LD | (MONHL),HL | | ;SAVE TEXT POINTER |
| 5C38 | ED7336FF | LD | (MONSP),SP | | ;SAVE STACK POINTER |
| | | | | | ;MONITOR COMMAND INPUT ROUTINE |
| 5C3C | 015E5C | MONBGN: | LD | BC, MONERR | |
| 5C3F | C5 | PUSH | BC | | |
| 5C40 | 3E2A | LD | A,*' | | ;MONITOR PROMPT MARK |
| 5C42 | CD5702 | CALL | CONOUT | | |
| 5C45 | CDE20B | CALL | DSPCSR | | |
| 5C48 | CDAD5F | CALL | MONCIN | | ;DISPLAY CURSOR AND INPUT COMMAND |
| 5C4B | 21875C | LD | HL, MONKTB | | |
| 5C4E | 010C00 | LD | BC, 12 | | ;NUMBER OF COMMAND |
| 5C51 | EDB1 | CPIR | | | |
| 5C53 | C0 | RET | NZ | | ;NOT COMMAND |
| 5C54 | 216F5C | LD | HL, MONATB | | |
| 5C57 | 09 | ADD | HL, BC | | |
| 5C58 | 09 | ADD | HL, BC | | |
| 5C59 | 5E | LD | E,(HL) | | ;CALCULATE ADDRESS |
| 5C5A | 23 | INC | HL | | ;AND JUMP TO COMMAND |
| 5C5B | 56 | LD | D,(HL) | | |
| 5C5C | EB | EX | DE, HL | | |
| 5C5D | E9 | JP | (HL) | | |
| 5C5E | CDCA5F | MONERR: | CALL | MONCLF | |
| 5C61 | 3E3F | LD | A,? | | ;MONITOR ERROR MESSAGE MARK |
| 5C63 | CD5702 | CALL | CONOUT | | |
| | | | | | ;MONITOR WARM START |
| 5C66 | ED7B36FF | MONHOT: | LD | SP,(MONSP) | |
| 5C6A | CDCA5F | CALL | MONCLF | | ;REPAIR STACK AND INPUT COMMAND |
| 5C6D | 18CD | JR | MONBGN | | |
| 5C6F | 665C | MONATB: | DW | MONHOT | ;ESC KEY |
| 5C71 | 935C | DW | MONCTB | | ;^B KEY |
| 5C73 | 665C | DW | MONHOT | | ;SPACE KEY |
| 5C75 | 665C | DW | MONHOT | | ;^J KEY |
| 5C77 | 665C | DW | MONHOT | | ;^M OR RETURN KEY |
| 5C79 | 3C5C | DW | MONBGN | | ;^L AND CLR KEY |
| 5C7B | E65D | DW | MONTM | | ;T KEY |
| 5C7D | 745D | DW | MONW | | ;W KEY |
| 5C7F | AE5D | DW | MONL | | ;L KEY |
| 5C81 | 685D | DW | MONGO | | ;G KEY |
| 5C83 | 165D | DW | MONDM | | ;D KEY |
| 5C85 | 995C | DW | MONSET | | ;S KEY |
| 5C87 | 5344474C | MONKTB: | DB | 'SDGLWT',0CH,0DH,0AH,'',2,1BH | |
| 5C8B | 57540C0D | | | | |
| 5C8F | 0A20021B | | | | |

CPI

を、BCレジスタ対のバイト・カウンタが0になってパリティ/オーバーフロー・フラグ(P/V)が0にリセットされるか、またはAレジスタの内容とHLレジスタ対で指定するメモリの内容が等しいためゼロ・フラグ(Z)が1にセットされるまでくり返す命令です。

同じ機能を他の命令で書き換えれば、

REPEAT: CPI

JR Z, EXIT

JP PE, REPEAT

EXIT:

などとなります。

このように、

CPIR

を用いれば、1ステップでメモリ上に置かれているデータ・ブロックの中にAレジスタの内容と同じデータが存在するか否か、また存在する場合のアドレスまでも調べる事ができます。

この命令を活用した良いプログラム例はないものかと考

えていたところ、なんとPC-8001の5C2CH番地から内蔵されているマシン語モニタで使っていた事に気が付きました。Z80特有の命令を全くと言って良い程使っていないN-BASICにしては非常に気の効いた使い方をしてしますので、システム解析の勉強にでもと思い掲載しておきます(第96図)。

このマシン語モニタでは、コマンド入力の際にキーボードから入力された文字がコマンドか否かの判断と、各コマンドの処理ルーチンのアドレスを得るために、

CPIR

を利用しています。

◎CPD

CPDは、

Compare and Decrement

の意味です。

CP I

と同じように、Aレジスタの内容とHLレジスタ対で指定するメモリの内容を比較して、比較結果をサイン・フラグ(S)およびゼロ・フラグ(Z)に表わしますが、比較後は、HLレジスタ対およびBCレジスタ対両者の内容共にディクリメントします。このディクリメントによってBCレジスタ対の内容が0(ゼロ)になればパリティ／オーバーフロー・フラグ(P/V)を0にリセットします。

他の命令に置き換えれば、

```
CP      A, (HL)
DEC     HL
DEC     BC
```

または、

```
CP I
DEC     HL
DEC     HL
```

などとなりますが、キャリ・フラグ(CY)に影響を及ぼさない等フラグの変化は独自のものです。

◎CPDR

CPDRは、

Compare, Decrement and Repeat

の意味で、

CPIR

と同じように、1ステップでメモリ上に置かれているデータ・ブロックの中にAレジスタの内容と同じデータが存在するか否か、また存在する場合のアドレスを調べる事のできる命令です。具体的には、

CPD

をBCレジスタ対が0(ゼロ)になるか、Aレジスタの内容と同じデータを見付けるまでくり返します。

〈第97図〉 ニーモニック↔マシン語対照表

ブロック転送

ブロック・サーチ

| | | | |
|------|----------|------|----------|
| LDI | ED A0 | CP I | ED A1 |
| LDIR | ED B0 | CRIR | ED B1 |
| LDD | ED A8 | CPD | ED A9 |
| DDR | ED B8 | CPDR | ED B9 |

ビット操作命令

ビット操作命令を使用する事によって、Aレジスタを含む全ての汎用レジスタおよびHLレジスタ対またはインデックス・レジスタで指定するメモリ上の任意のビットに対して、セットまたはリセットを行う事が可能で、そのビットが0であるか1であることを調べる事もできます。

8080CPU当時のビット操作は専用の命令がなかったために、論理演算命令やローテート・シフト命令の組み合わせのみで操作を行わなければならず、非常に読みにくいプログラムを組んでいました。おかげで論理演算の勉強にはなりましたが……。

さて、ビット操作命令のニーモニックには次の3種類がありますので対照表を御覧ください。

◎SET … ビットを1にセットする命令。

◎RES … ビットを0にリセットする命令。

◎BIT … ビットの内容を調べる命令。

なお、ビットの内容を調べた結果が、0であればゼロ・フラグ (Z) が1にセットされ、結果が1であればゼロ・フラグ (Z) が0にリセットされます。

まとめ

第2ブロックから第3ブロックにわたり、長い間Z80のインストラクションを紹介して来ましたが、本章をもって全ての命令を説明し終わりました。

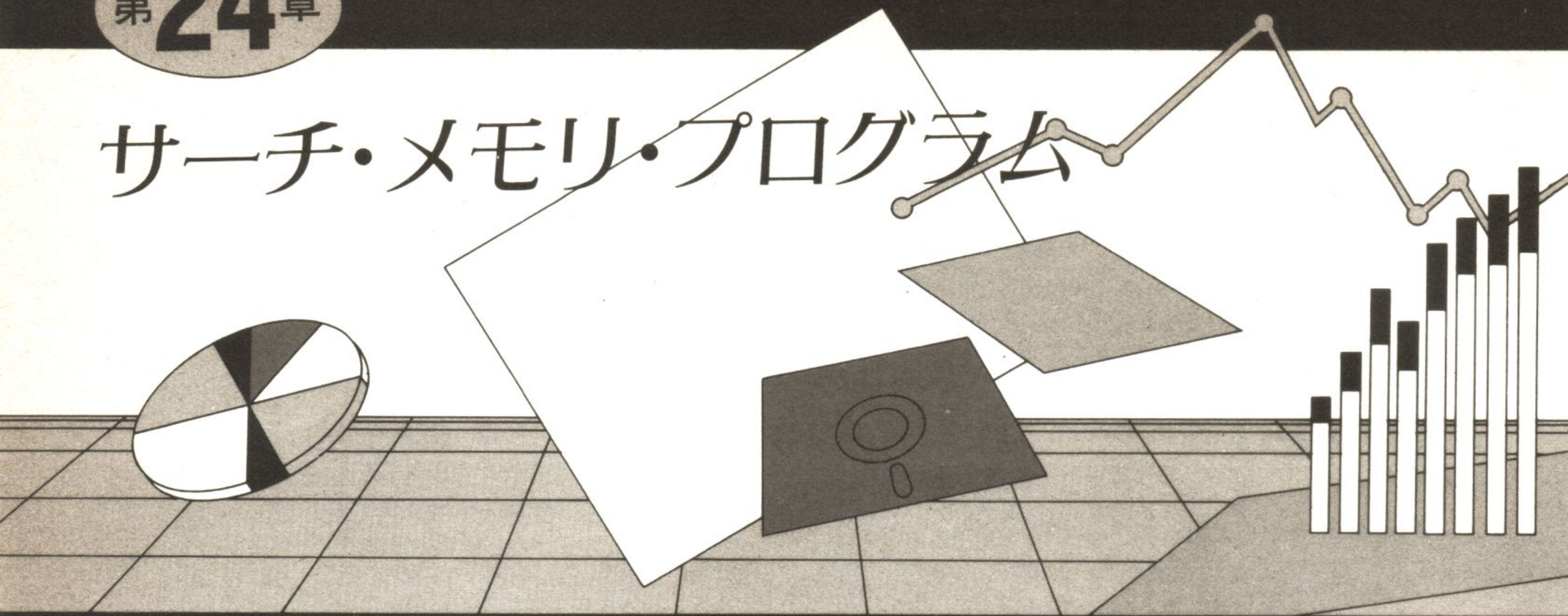
そこで、次章では総集編という事で少し長めのプログラムをオール・アセンブラで組んでみたいと思います。

〈第98図〉 ニーモニク↔マシン語対照表

ビット操作

| × | A | B | C | D | E | F | L | (HL) | (IX +d) | (IY +d) |
|----------|------------|------------|------------|------------|------------|------------|------------|------------|-------------------|-------------------|
| BIT 0, × | C B 4 7 | C B 4 0 | C B 4 1 | C B 4 2 | C B 4 3 | C B 4 4 | C B 4 5 | C B 4 6 | D D C B d 6 | F D C B d 6 |
| BIT 1, × | C B 4 F | C B 4 8 | C B 4 9 | C B 4 A | C B 4 B | C B 4 C | C B 4 D | C B 4 E | D D C B d E | F D C B d E |
| BIT 2, × | C B 5 7 | C B 5 0 | C B 5 1 | C B 5 2 | C B 5 3 | C B 5 4 | C B 5 5 | C B 5 6 | D D C B d 6 | F D C B d 6 |
| BIT 3, × | C B 5 F | C B 5 8 | C B 5 9 | C B 5 A | C B 5 B | C B 5 C | C B 5 D | C B 5 E | D D C B d E | F D C B d E |
| BIT 4, × | C B 6 7 | C B 6 0 | C B 6 1 | C B 6 2 | C B 6 3 | C B 6 4 | C B 6 5 | C B 6 6 | D D C B d 6 | F D C B d 6 |
| BIT 5, × | C B 6 F | C B 6 8 | C B 6 9 | C B 6 A | C B 6 B | C B 6 C | C B 6 D | C B 6 E | D D C B d E | F D C B d E |
| BIT 6, × | C B 7 7 | C B 7 0 | C B 7 1 | C B 7 2 | C B 7 3 | C B 7 4 | C B 7 5 | C B 7 6 | D D C B d 6 | F D C B d 6 |
| BIT 7, × | C B 7 F | C B 7 8 | C B 7 9 | C B 7 A | C B 7 B | C B 7 C | C B 7 D | C B 7 E | D D C B d E | F D C B d E |
| RES 0, × | C B 8 7 | C B 8 0 | C B 8 1 | C B 8 2 | C B 8 3 | C B 8 4 | C B 8 5 | C B 8 6 | D D C B d 6 | F D C B d 6 |
| RES 1, × | C B 8 F | C B 8 8 | C B 8 9 | C B 8 A | C B 8 B | C B 8 C | C B 8 D | C B 8 E | D D C B d E | F D C B d E |
| RES 2, × | C B 9 7 | C B 9 0 | C B 9 1 | C B 9 2 | C B 9 3 | C B 9 4 | C B 9 5 | C B 9 6 | D D C B d 6 | F D C B d 6 |
| RES 3, × | C B 9 F | C B 9 8 | C B 9 9 | C B 9 A | C B 9 B | C B 9 C | C B 9 D | C B 9 E | D D C B d E | F D C B d E |
| RES 4, × | C B A 7 | C B A 0 | C B A 1 | C B A 2 | C B A 3 | C B A 4 | C B A 5 | C B A 6 | D D C B d 6 | F D C B d 6 |
| RES 5, × | C B A F | C B A 8 | C B A 9 | C B A A | C B A B | C B A C | C B A D | C B A E | D D C B d E | F D C B d E |
| RES 6, × | C B B 7 | C B B 0 | C B B 1 | C B B 2 | C B B 3 | C B B 4 | C B B 5 | C B B 6 | D D C B d 6 | F D C B d 6 |
| RES 7, × | C B B F | C B B 8 | C B B 9 | C B B A | C B B B | C B B C | C B B D | C B B E | D D C B d E | F D C B d E |
| SET 0, × | C B C 7 | C B C 0 | C B C 1 | C B C 2 | C B C 3 | C B C 4 | C B C 5 | C B C 6 | D D C B d 6 | F D C B d 6 |
| SET 1, × | C B C F | C B C 8 | C B C 9 | C B C A | C B C B | C B C C | C B C D | C B C E | D D C B d E | F D C B d E |
| SET 2, × | C B D 7 | C B D 0 | C B D 1 | C B D 2 | C B D 3 | C B D 4 | C B D 5 | C B D 6 | D D C B d 6 | F D C B d 6 |
| SET 3, × | C B D F | C B D 8 | C B D 9 | C B D A | C B D B | C B D C | C B D D | C B D E | D D C B d E | F D C B d E |
| SET 4, × | C B E 7 | C B E 0 | C B E 1 | C B E 2 | C B E 3 | C B E 4 | C B E 5 | C B E 6 | D D C B d 6 | F D C B d 6 |
| SET 5, × | C B E F | C B E 8 | C B E 9 | C B E A | C B E B | C B E C | C B E D | C B E E | D D C B d E | F D C B d E |
| SET 6, × | C B F 7 | C B F 0 | C B F 1 | C B F 2 | C B F 3 | C B F 4 | C B F 5 | C B F 6 | D D C B d 6 | F D C B d 6 |
| SET 7, × | C B F F | C B F 8 | C B F 9 | C B F A | C B F B | C B F C | C B F D | C B F E | D D C B d E | F D C B d E |

サーチ・メモリ・プログラム



プログラムの概要

第3ブロックをまとめるプログラムの一つとして紹介するのが、サーチメモリ・プログラムです。

このプログラムは、メモリ上のデータの中から特定のデータ群(文字列)をさがし出すユーティリティ・プログラムで、私がN-BASICを解析した時ROM内のサブルーチンやデータ領域をさがし出すのに非常な助けになった、4バイトサーチ・プログラム(橋本正樹氏原作)からアイデアを得てくったものです。

先の橋本氏のプログラムでは、メインルーチンはN-BASICで組まれており、最も高速性を要するサーチルーチンのみが、マシン語のサブルーチンとして用意されていました。キーボードからの入力や、CRTスクリーンへの表示等は、全てN-BASICで管理していたのですが、実用上、十分過ぎるくらいに十分な速度で24KバイトのROMをサーチし終えるのに1秒もかからなかったと記憶しています。

さて、これから紹介するサーチメモリプログラムは、全てアセンブラで組んでみました。N-BASICとリンクさせても十分な速度が得られるものなのですが、面倒なキー入力およびCRTスクリーンへの表示が行いたかったため、あえてアセンブラのみで組んだのです。

したがってこのプログラムは、前章までに紹介

して来た何本かのプログラム例と技術的にかなりの差をつけてある事になります。

だからと言ってあまり難しく考える必要はありません。ここまで本書を読んで来た皆さんの気力が有れば、それだけでマシン語をマスターするに十分です。

プログラムの仕様

プログラミング(コーディング)を行う前に必ず仕様を決めなくてはなりません。プログラムには必ず目的があるはずで、その目的を実現するためのいろいろな取り決めが仕様です。

たとえば、スペースインベーダ・ゲームの目的はインベーダ・ゲームが楽しめる事で、

インベーダの数
インベーダの動き方
砲台の動き方
キー操作の方法
etc.

などは、すべてプログラムの仕様として決めておかねばなりません。BASIC等高級言語の場合には、コーディング途中で仕様を変更する事も比較的容易ですが(もちろん根本的な変更は論外です)、マシン語で特にハンドアSEMBルを行う場合などは、メッセージ一つを変更するのも予想以上に困難な場合が多いので、最初の頭の中や紙の上

で仕様を決める時によく注意しなくてはなりません。

それでは、「サーチメモリ・プログラム」の目的は何でしょうか？

目的は、メモリの中から任意のデータ群をさがし出す（サーチする）事です。そのためには、任意のデータ群（キーワード）を入力するためのルーチンも必要ですしサーチ結果を表示するルーチンも、もちろんサーチも行わなければいけません。

〈第101図〉 プロンプトメッセージ“Ok”をサーチ

```
def usr=&hc100:a=usr(a)
```

```
code — "O
code — "k
code —
3B60
Ok
```

```
def usr=&hc100:a=usr(a)
```

```
code — "O
code — "k
code — fc
code — 0d
code — 0a
code — 00
code —
3B60
Ok
```

また、サーチするメモリ領域は、0000H番地から5FFFH番地のROM領域としておきますが、これは他機種に移植する場合などのために簡単に変更できるようにしておきます。

Z80の命令でサーチする場合と、文字列でサーチする場合を考慮して、キーワードは16進数2桁でもキャラクタでも入力できる様にします。ただし、キャラクタをキーワードとして入力する場合には頭にダブルクォート（"）を付ける事にします。また、キーワードを入力せずにRETURNキーのみが入力された場合には、キーワードの入力を終了した事になります。

キーワードの数（文字列の長さ）は、プログラムの組み易さから考えて255バイトまで可能とし、キーワードが1個も存在しなかった場合にはサーチを行わない様にします。

プログラム自体はサブルーチンの形をとり、N-BASICのUSR関数として実行できるようにしました。

以上の仕様に基づいてつくったサーチメモリ・プログラムのアセンブルリストを第99図として、また打ち込みたい方のためにダンプリストを第100図として掲載します。

さらに、サーチメモリの実行例として、N-BASICのプロンプトメッセージである“Ok”をサーチした例を第101図として、Z80の無条件絶対ジャンプ命令（C3H）をサーチした例を第102図として示します。

〈第102図〉 Z80の絶対ジャンプ命令（C3H）をサーチ

```
def usr=&hc100:a=usr(a)
```

```
code — c3
code —
0004 0008 000B 0010 0013 0018 001D 0020 0023 0028 002B 0030 0035 0038 007E 0096
019D 01E6 02AD 0366 0477 04AC 0642 06E7 06EC 0773 0784 0881 09F9 0AE6 0AF6 0C19
0C96 0CD4 0CE9 0D31 0D5D 0DA8 0DB5 0F19 0F72 12D6 14A7 159C 15C3 15FD 178D 181D
182F 1877 188E 18E9 191A 1A2D 1A4D 1A68 1A70 1A74 1AB3 1E6D 1F9F 1FB3 1FC6 20E9
2127 212D 21ED 21F6 224C 225F 22D2 22DB 22DE 2361 236C 2373 237C 23A6 23B2 23ED
24A1 24AE 25FD 25FE 2635 264C 266E 268D 27CD 27DC 27F7 27FE 283C 28A8 28FA 2927
293F 294D 2984 299A 29AF 2A1F 2A46 2B1B 2B22 2B87 2B88 2BB4 2C45 2CB8 2CF5 2D08
2D1F 2E66 2EBB 2F07 2F65 3196 3230 32E0 333D 336F 34FF 3595 3664 36FF 37D5 3B29
3B2C 3BC3 3BD6 3C13 3C42 3CD1 3CE9 3D73 3E62 3EE5 3EF0 3F44 3F80 3FAC 3FCC 3FCF
3FE2 4002 4035 4055 4061 407F 40A3 40E7 427D 42A3 42F0 42FF 432C 4369 4380 4452
44A7 453A 4569 45A0 466D 46C1 46CC 46FF 4726 473F 47C6 47FC 4812 4862 48A9 49A0
49D4 4A05 4A70 4A88 4B29 4BF6 4C27 4DB4 4DD5 4EF0 4FBF 5071 507F 50EF 519E 525F
526D 52E9 5397 540E 5445 54B5 5526 56AA 56B0 576B 57D8 57DE 57F0 5849 5866 58C7
58D6 5915 5943 5A98 5ADF 5AE8 5B3C 5B7D 5BBF 5BED 5C29 5D0F 5DAB 5DDB 5DE3 5E0F
5E80 5F9B 5FD1 5FEA
Ok
```


〈第99図〉 サーチメモリプログラム

| | | | |
|---------|-------------|--|-----------------------------------|
| 《Aブロック》 | | ; PC-8001 SEARCH MEMORY PROGRAM Ver 1.0 | |
| | | ; Copyright 1982 (C) by KIYOSHI KAWAMURA | |
| | | ; in FORESIGHT PLANNING SECTION | |
| | | ; | |
| 0 2 5 7 | | DSPCHR: EQU 0 2 5 7 H | ; display a character |
| 0 F 7 5 | | INCHR: EQU 0 F 7 5 H | ; input a character from keyboard |
| | | ; | |
| 0 0 0 0 | | ADRTOP: EQU 0 0 0 0 H | ; top address of search area |
| 5 F F F | | ADREND: EQU 5 F F F H | ; end address of search area |
| | | ; | |
| | | ORG 0 C 1 0 0 H | |
| 《Bブロック》 | | ; input key-words section | |
| | | ; | |
| C 1 0 0 | 2 1 F 7 C 1 | LD HL, TABLE | |
| C 1 0 3 | 0 E 0 0 | LD C, 0 | ; clear key-word counter |
| | | ; | |
| C 1 0 5 | 1 1 E 9 C 1 | NWORD: LD DE, PROMPT | |
| C 1 0 8 | C D E 0 C 1 | CALL DSPMSG | ; display the prompt message |
| | | ; | |
| C 1 0 B | C D 6 E C 1 | CALL INCOD | |
| C 1 0 E | 0 5 | DEC B | |
| C 1 0 F | 2 8 1 E | JR Z, EIN | ; end of all input key-word |
| | | ; | |
| C 1 1 1 | 1 0 0 3 | DJNZ TWOCLM | |
| C 1 1 3 | 5 A | LD E, D | |
| C 1 1 4 | 1 6 3 0 | LD D, 0 | ; clear msb byte of key-word |
| | | ; | |
| C 1 1 6 | 7 A | TWOCLM: LD A, D | |
| C 1 1 7 | F E 2 2 | CP | |
| C 1 1 9 | 7 B | LD A, E | |
| C 1 1 A | 2 8 0 D | JR Z, EWORD | |
| | | ; | |
| C 1 1 C | C D B 4 C 1 | CALL HEXBIN | ; convert msb byte into binary |
| C 1 1 F | 4 7 | LD B, A | |
| | | ; | |
| C 1 2 0 | 7 A | LD A, D | |
| C 1 2 1 | C D B 4 C 1 | CALL HEXBIN | ; convert 1 sb byte into binary |
| | | ; | |
| C 1 2 4 | 0 7 | RLCA | |
| C 1 2 5 | 0 7 | RLCA | |
| C 1 2 6 | 0 7 | RLCA | |
| C 1 2 7 | 0 7 | RLCA | |
| C 1 2 8 | 8 0 | ADD A, B | |
| | | ; | |
| C 1 2 9 | 7 7 | EWORD: LD (HL), A | ; store a key-word |
| C 1 2 A | 2 3 | INC HL | |
| C 1 2 B | 0 C | INC C | ; increment byte counter |
| C 1 2 C | D 8 | RET C | ; number of key-words overflow |
| C 1 2 D | 1 8 D 6 | JR NWORD | |
| | | ; | |
| C 1 2 F | 3 E 0 D | EIN: LD A, 0 D H | |
| C 1 3 1 | C D 5 7 0 2 | CALL DSPCHR | ; display a carriage return code |
| C 1 3 4 | 3 E 0 A | LD A, 0 A H | |
| C 1 3 6 | C D 5 7 0 2 | CALL DSPCHR | ; display a line feed code |
| | | ; | |
| C 1 3 9 | 0 C | INC C | |
| C 1 3 A | 0 D | DEC C | |
| C 1 3 B | C 8 | RET Z | ; no key-word, return to basic |
| 《Cブロック》 | | ; search main section | |
| | | ; | |
| C 1 3 C | 2 1 0 0 0 0 | LD HL, ADRTOP | ; set top address of search area |
| C 1 3 F | 1 1 F F 5 F | IFEND: LD DE, ADREND | ; set end address of search area |
| | | ; | |
| C 1 4 2 | E 5 | PUSH HL | |
| C 1 4 3 | A 7 | AND A | ; reset carry flag |
| C 1 4 4 | E D 5 2 | SBC HL, DE | |
| C 1 4 6 | E 1 | POP HL | |
| C 1 4 7 | C 8 | RET Z | ; end of search, return to basic |
| | | ; | |
| C 1 4 8 | C 5 | PUSH BC | |
| C 1 4 9 | E 5 | PUSH HL | |

| | | | | | |
|---------|-------------|----------|------|------------|--|
| C 1 4 A | 1 1 F 7 C 1 | | LD | DE, TABLE | |
| C 1 4 D | 1 A | MLOOP : | LD | A, (DE) | |
| C 1 4 E | BE | | CP | (HL) | |
| C 1 4 F | 2 0 0 7 | | JR | NZ, NXTMEM | ; not same, next memory |
| | | | | | |
| C 1 5 1 | 0 D | | DEC | C | |
| C 1 5 2 | 2 8 0 4 | | JR | Z, NXTMEM | ; end of key-words, next memory |
| | | | | | |
| C 1 5 4 | 1 3 | | INC | DE | |
| C 1 5 5 | 2 3 | | INC | HL | |
| C 1 5 6 | 1 8 F 5 | | JR | MLOOP | |
| | | | | | |
| C 1 5 8 | E 1 | NXTMEM : | POP | HL | |
| C 1 5 9 | C 1 | | POP | BC | |
| C 1 5 A | 2 3 | | INC | HL | |
| C 1 5 B | 2 0 E 2 | | JR | NZ, IFEND | |
| | | | | | |
| C 1 5 D | 2 B | FOUND : | DEC | HL | |
| C 1 5 E | 7 C | | LD | A, H | |
| C 1 5 F | C D C 0 C 1 | | CALL | DSPACC | ; display a msb byte of address |
| C 1 6 2 | 7 D | | LD | A, L | |
| C 1 6 3 | C D C 0 C 1 | | CALL | DSPACC | ; display a lsb byte of address |
| C 1 6 6 | 3 E 2 0 | | LD | A, | |
| C 1 6 8 | C D 5 7 0 2 | | CALL | DSPCHR | ; display a space code |
| C 1 6 B | 2 3 | | INC | HL | |
| C 1 6 C | 1 8 D 1 | | JR | IFEND | |
| | | | | | |
| 《Dブロック》 | | | | | ; input a key-word subrouthne |
| | | | | | |
| | | | | | outputs : input character in register de |
| | | | | | : input clumn plus one in register c |
| | | | | | destorys : register a,f,b,d,e |
| C 1 6 E | 0 6 0 1 | INCOD : | LD | B, 1 | |
| C 1 7 0 | C D 7 5 0 F | | CALL | INCHR | |
| C 1 7 3 | F E 0 D | | CP | 0 DH | |
| C 1 7 5 | C 8 | | RET | Z | ; return to main routine |
| | | | | | |
| C 1 7 6 | F E 7 F | | CP | 7 FH | |
| C 1 7 8 | 2 8 F 4 | | JR | Z, INCOD | |
| | | | | | |
| C 1 7 A | F E 2 0 | | CP | 2 0 H | |
| C 1 7 C | 3 8 F 0 | | JR | C, INCOD | ; cancel control code |
| | | | | | |
| C 1 7 E | C D 5 7 0 2 | | CALL | DSPCHR | ; echo-back a input character |
| C 1 8 1 | 5 7 | | LD | D, A | |
| | | | | | |
| C 1 8 2 | 0 6 0 2 | INCOD1 : | LD | B, 2 | |
| C 1 8 4 | C D 7 5 0 F | | CALL | INCHR | |
| C 1 8 7 | F E 0 D | | CP | 0 DH | |
| C 1 8 9 | C 8 | | RET | Z | ; return to main routine |
| | | | | | |
| C 1 8 A | F E 0 8 | | CP | 0 8 H | |
| C 1 8 C | 2 8 0 4 | | JR | Z, DEL1 | |
| C 1 8 E | F E 7 F | | CP | 7 FH | |
| C 1 9 0 | 2 0 0 5 | | JR | NZ, INCOD2 | |
| | | | | | |
| C 1 9 2 | C D D 7 C 1 | DEL1 : | CALL | BACK | ; display a back-space |
| C 1 9 5 | 1 8 D 7 | | JR | INCOD | |
| | | | | | |
| C 1 9 7 | F E 2 0 | INCOD2 : | CP | 2 0 H | |
| C 1 9 9 | 3 8 E 7 | | JR | C, INCOD1 | ; cancel control code |
| | | | | | |
| C 1 9 B | C D 5 7 0 2 | | CALL | DSPCHR | ; echo-back a input caracter |
| C 1 9 E | 5 F | | LD | E, A | |
| | | | | | |
| C 1 9 F | 0 6 0 3 | INCOD3 : | LD | B, 3 | |
| C 1 A 1 | C D 7 5 0 F | | CALL | INCHR | |
| C 1 A 4 | F E 0 D | | CP | 0 DH | |
| C 1 A 6 | C 8 | | RET | Z | ; return to main routine |
| | | | | | |
| C 1 A 7 | F E 0 8 | | CP | 0 8 H | |
| C 1 A 9 | 2 8 0 4 | | JR | Z, DEL2 | |

| | | | | |
|--|----------|---------|-----------------------|-----------------------|
| C1AB | FE7F | CP | 7FH | |
| C1AD | 20F0 | JR | NZ,INCOD3 | |
| ; | | | | |
| C1AF | CDD7C1 | DEL2: | CALL BACK | ;display a back-space |
| C1B2 | 18CE | | JR INCOD1 | |
| ; | | | | |
| ; convert hexa corde into binary code subroutine | | | | |
| 《Eブロック》 | | | | |
| ; | | | | |
| ; inputs : character code in accumulator | | | | |
| ; outputs : binary code in accumulator | | | | |
| ; destroys : register a,f | | | | |
| ; | | | | |
| C1B4 | FE3A | HEXBIN: | CP '9'+1 | |
| C1B6 | 3805 | | JR C,HEXBI1 | |
| C1B8 | E60F | | AND 0FH | |
| C1BA | C609 | | ADD A,9 | |
| C1BC | C9 | | RET | |
| ; | | | | |
| C1BD | E60F | HEXBI1: | AND 0FH | |
| C1BF | C9 | | RET | |
| ; | | | | |
| ; display accumulator hexa decimal subroutine | | | | |
| 《Fブロック》 | | | | |
| ; | | | | |
| ; destroys : register a,f,b | | | | |
| ; | | | | |
| C1C0 | 47 | DSPACC: | LD B,A | |
| C1C1 | 0F | | RRCA | |
| C1C2 | 0F | | RRCA | |
| C1C3 | 0F | | RRCA | |
| C1C4 | 0F | | RRCA | |
| C1C5 | CDC9C1 | | CALL HALF | |
| C1C8 | 78 | | LD A,F | |
| ; | | | | |
| C1C9 | E60F | HALF: | AND 0FH | |
| C1CB | FE0A | | CP 10 | |
| C1CD | 3802 | | JR C,NUMBER | |
| C1CF | C607 | | ADD A,7 | |
| ; | | | | |
| C1D1 | C630 | NUMBER: | ADD A,'0' | |
| C1D3 | CD5702 | | CALL DSPCHR | |
| C1D6 | C9 | | RET | |
| ; | | | | |
| ; display a back-space subroutine | | | | |
| 《Gブロック》 | | | | |
| ; | | | | |
| ; destroys : register a,f | | | | |
| ; | | | | |
| C1D7 | D5 | BACK: | PUSH DE | |
| C1D8 | 11F3C1 | | LD DE,BACKSPC | |
| C1DB | CDE0C1 | | CALL DSPMSG | |
| C1DE | D1 | | POP DE | |
| C1DF | C9 | | RET | |
| ; | | | | |
| ; display message subroutine | | | | |
| 《Hブロック》 | | | | |
| ; | | | | |
| ; destroys : register a,f,d,e | | | | |
| ; | | | | |
| C1E0 | 1A | DSPMSG: | LD A,(DE) | |
| C1E1 | A7 | | AND A | |
| C1E2 | C8 | | RET Z | |
| C1E3 | CD5702 | | CALL DSPCHR | |
| C1E6 | 13 | | INC DE | |
| C1E7 | 18F7 | | JR DSPCHR | |
| ; | | | | |
| ; data area | | | | |
| 《Iブロック》 | | | | |
| C1E9 | 0D0A636F | PROMPT: | DEFB 0DH,0AH,'code',0 | |
| C1ED | 6465202D | | | |
| C1F1 | 2000 | | | |
| C1F3 | 1D201D00 | BAKSPC: | DEFB 1DH,' ',1DH,0 | |
| ; | | | | |
| ; key-word table | | | | |
| 《Jブロック》 | | | | |
| C1F7 | | TABLE: | END | |

〈第100図〉サーチ・メモリプログラム (ダンプリスト)

| | | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------------------|
| C100 | 21 | F7 | C1 | 0E | 00 | 11 | E9 | C1 | CD | E0 | C1 | CD | 6E | C1 | 05 | 28 | !.....n..(|
| C110 | 1E | 10 | 03 | 5A | 16 | 30 | 7A | FE | 22 | 7B | 28 | 0D | CD | B4 | C1 | 47 | ...Z.0z."{(....G |
| C120 | 7A | CD | B4 | C1 | 07 | 07 | 07 | 07 | 80 | 77 | 23 | 0C | D8 | 18 | D6 | 3E | z.....w#....> |
| C130 | 0D | CD | 57 | 02 | 3E | 0A | CD | 57 | 02 | 0C | 0D | C8 | 21 | 00 | 00 | 11 | ..W.>..W.....!... |
| C140 | FF | 5F | E5 | A7 | ED | 52 | E1 | C8 | C5 | E5 | 11 | F7 | C1 | 1A | BE | 20 | —...R..... |
| C150 | 07 | 0D | 28 | 04 | 13 | 23 | 18 | F5 | E1 | C1 | 23 | 20 | E2 | 2B | 7C | CD | ..(..#.....#..+!.. |
| C160 | C0 | C1 | 7D | CD | C0 | C1 | 3E | 20 | CD | 57 | 02 | 23 | 18 | D1 | 06 | 01 |>..W.#..... |
| C170 | CD | 75 | 0F | FE | 0D | C8 | FE | 7F | 28 | F4 | FE | 20 | 38 | F0 | CD | 57 | ..u.....(....8..W |
| C180 | 02 | 57 | 06 | 02 | CD | 75 | 0F | FE | 0D | C8 | FE | 08 | 28 | 04 | FE | 7F | ..W...u.....(.... |
| C190 | 20 | 05 | CD | D7 | C1 | 18 | D7 | FE | 20 | 38 | E7 | CD | 57 | 02 | 5F | 06 |8..W.—. |
| C1A0 | 03 | CD | 75 | 0F | FE | 0D | C8 | FE | 08 | 28 | 04 | FE | 7F | 20 | F0 | CD | ..u.....(.... |
| C1B0 | D7 | C1 | 18 | CE | FE | 3A | 38 | 05 | E6 | 0F | C6 | 09 | C9 | E6 | 0F | C9 |:8..... |
| C1C0 | 47 | 0F | 0F | 0F | 0F | CD | C9 | C1 | 78 | E6 | 0F | FE | 0A | 38 | 02 | C6 | G.....x.....8.. |
| C1D0 | 07 | C6 | 30 | CD | 57 | 02 | C9 | D5 | 11 | F3 | C1 | CD | E0 | C1 | D1 | C9 | ..0..W..... |
| C1E0 | 1A | A7 | C8 | CD | 57 | 02 | 13 | 18 | F7 | 0D | 0A | 63 | 6F | 64 | 65 | 20 |W.....code |
| C1F0 | 2D | 20 | 00 | 1D | 20 | 1D | 00 | | | | | | | | | | — |

それでは、各ブロックごとにアセンブルをリスト追ってみましょう。

◎Aブロック

本プログラムでは、2種類のシステムサブルーチンを利用しています。

一つは、Aレジスタのキャラクタコードに基づいたキャラクタの表示を行う0257H番地のサブルーチンもう一つはキーボードから入力されたキャラクタのコードをアキュムレータ(Aレジスタ)に与えてもどる0F75H番地のサブルーチンです。

「移植し易い様にシステムサブルーチンは使わない」という思想を軸にしているのですが、この最も基本的でかつ複雑なIOCS (Input Output Control Subroutine) は、ハードウェアへの依存度が非常に高く、また、それらのサブルーチンは全てのパーソナルコンピュータに用意されていると言っても過言ではないと思い二つのシステムサブルーチンを利用しました。

他の機種に移植する場合には、二つのシンボル定義擬似命令(EQU)のオペランドを各機種用に変更してください。またCRT画面への表示を行う0257H番地のかわりにプリンタ出力サブルーチンなどを使ってもおもしろいと思います。

◎Bブロック

プログラムを実行すると最初にキーワードの入力を行います。

このルーチンを通過する事によって、入力した

キーワードがキーワード・テーブルに格納され、またCレジスタにキーワードのバイト数が与えられます。

BASIC等と異なりアセンブラでは、このような作業に意外と手間がかかるもので、本プログラムでもメインのサーチルーチンよりキー入力ルーチンの方にメモリを使用しています。

キー入力ルーチンの最後では、キーワードのバイト数が0であるか否かの判定を行っており0であった場合には次のサーチルーチンへ行かずにN—BASICにもどります。

◎Cブロック

本プログラムのメインであるサーチルーチンを見て何人かの皆さんは「どこかで見た事があるな?」と思われたかも知れません。

実はこのルーチンは、第3ブロックの第19章で紹介した野本正司さんから送っていただいた「N—BASICのプロンプト・メッセージをさがし出すプログラム」をほんの少し変更しただけのものなのです。

「MLOOP」というラベルのついているブロックは、全てのキーワードが合っていればゼロフラグ(Z)をセットし、途中でひとつでも異なっていればゼロフラグ(Z)をリセットして「NXTMEM」のラベルにジャンプします。

「FOUND」の部分では、サーチされたアドレスを4桁の16進数として表示しています。

☆Dブロック

「INCOD」は、サーチメモリ・プログラムの中で一番長いサブルーチンで、動作も比較的複雑です。

キーボードからサーチするためのキーワードを入力するためのサブルーチンで、N-BASIC内蔵モニタのように簡略化すれば10ステップ程度にできたのですが、あまり短くてはおもしろくありませんし使い勝手も良く無いと思い改良して行くうちに、プリンタ用紙で1ページ近いサブルーチンになってしまいました。

このサブルーチンをコールする事によって、キーボードから2バイトのキャラクタを入力し、それぞれのキャラクタコードをDレジスタおよびEレジスタに、格納してもどります。またCレジスタには、キーボードから入力されたキャラクタのバイト数に1を加えたもの(1~3)が入ります。

コードが20H未満のコントロールコードが入力された場合は無視されますが、08Hおよび7FHのバックスペースコードでは、一文字前に入力されたキャラクタを無効にしています。

☆Eブロック

「HEXBIN」は、Aレジスタに与えられた16進関係のキャラクタコード(0~9およびA~F)を対応する2進データ(バイナリコード)に変換するためのサブルーチンです。

☆Fブロック

「DSPACC」は、Aレジスタのデータを2桁の16進数としてCRTスクリーンに表示するためのサブルーチンです。

バイナリコードを16進のキャラクタに変換する良い例では無いでしょうか？

☆Gブロック

バックスペース表示サブルーチン(「BACK」)では、バックスペース用のメッセージ・データを表示します。

具体的には、バックスペース用データが格納してあるアドレス(「BAKSPC」)をDEレジスタ対に入れて、メッセージ表示サブルーチン(「DSP

MSG」)をコールしています。

☆Hブロック

「DSPMSG」とラベルを付けたメッセージ表示サブルーチンでは、CRTスクリーンに文字列(メッセージ)を表示します。

このサブルーチンをコールする場合には、DEレジスタ対に文字列データの格納してある先頭アドレスが入っていないてはいけません。

「DSPMSG」では、このDEレジスタ対をインクリメントしながらメモリ上のデータをAレジスタにとり込み、N-BASIC内の一文字表示サブルーチン(「DSPCHR」)を利用して文字列の表示を行います。

メモリ上のデータが00H(エンドマーク)であればメインルーチンにもどります。

普通このような場合には、ポインタとしてHLレジスタ対を使用するのですが、ここでは丁度DEレジスタ対が余っていたためにDEレジスタ対を使用しました。ちなみにN-BASICのメッセージ出力サブルーチン(52EDH番地)では、HLレジスタ対がポインタとなっています。

☆Iブロック

この、「サーチメモリ・プログラム」では、2種類のメッセージをデータとして用意しています。

一つは、キーワードを入力する時にキーボードからの入力をうながすために表示するプロンプト・メッセージで、改行するための、

0DH (キャリッジリターン・コード)

0AH (ラインフィード・コード)

の後に、

code —

のキャラクタコード、さらにメッセージの末尾である事を示す

00H (メッセージ・エンドマーク)

で構成されています。0DHおよび0AHについてはN-BASICによって決められているものですが、他の機種でも同じように、0DHと0AHを表示する事によって改行を行う機種(インタプリタ)が多い様です。ただし、一部の機種では0DHのみで改行するもの等、例外もありますので、

その辺の詳細は各機種のキャラクタコード表（機能コード表）などを調べてみてください。

また00Hをメッセージ・エンドマークとして使う事に関しては、本プログラムのメッセージ表示サブルーチンの仕様がそうになっているために00Hを使用しているだけで、深い意味はありません。たとえば、FFHをエンドマークとして使いたければ、メッセージ表示サブルーチン（「DSP MSG」とラベルの付いているサブルーチン）を変更すれば良いのです。

しかし、このエンドマークもほとんどの機種（インタプリタ等）で00Hを使っている様です（N-BASICも00Hを使用）。

二つめのメッセージデータは、キーワードを入力する際の入力ミスで「DEL」または「CTRL+H」のキーを押した場合に、カーソルの前のキャラクタを一文字消去するためのデータです。

データは、1DH、20H、1DHおよびエンドマーク（00H）の4バイトから構成されていますが、この20Hは、御存知のようにスペース（空白）のコード、そして1DHはカーソルを左に動かす（LEFT ARROW）ためのコードです。

つまり、始めの1DH（LEFT ARROW）でカーソルを1キャラクタ分左へもどし、20Hのスペース（空白）を表示して元のキャラクタを消去し、スペース（空白）を表示したために右に移動したカーソルを、再度1DH（LEFT ARROW）によってもどす事によって、バックスペース（BA

CK SPACE）を実現するのです（第103図）。

☆Jブロック

キーワードを格納しておくためのキーワードテーブルは、プログラム末から用意しました。

移植について

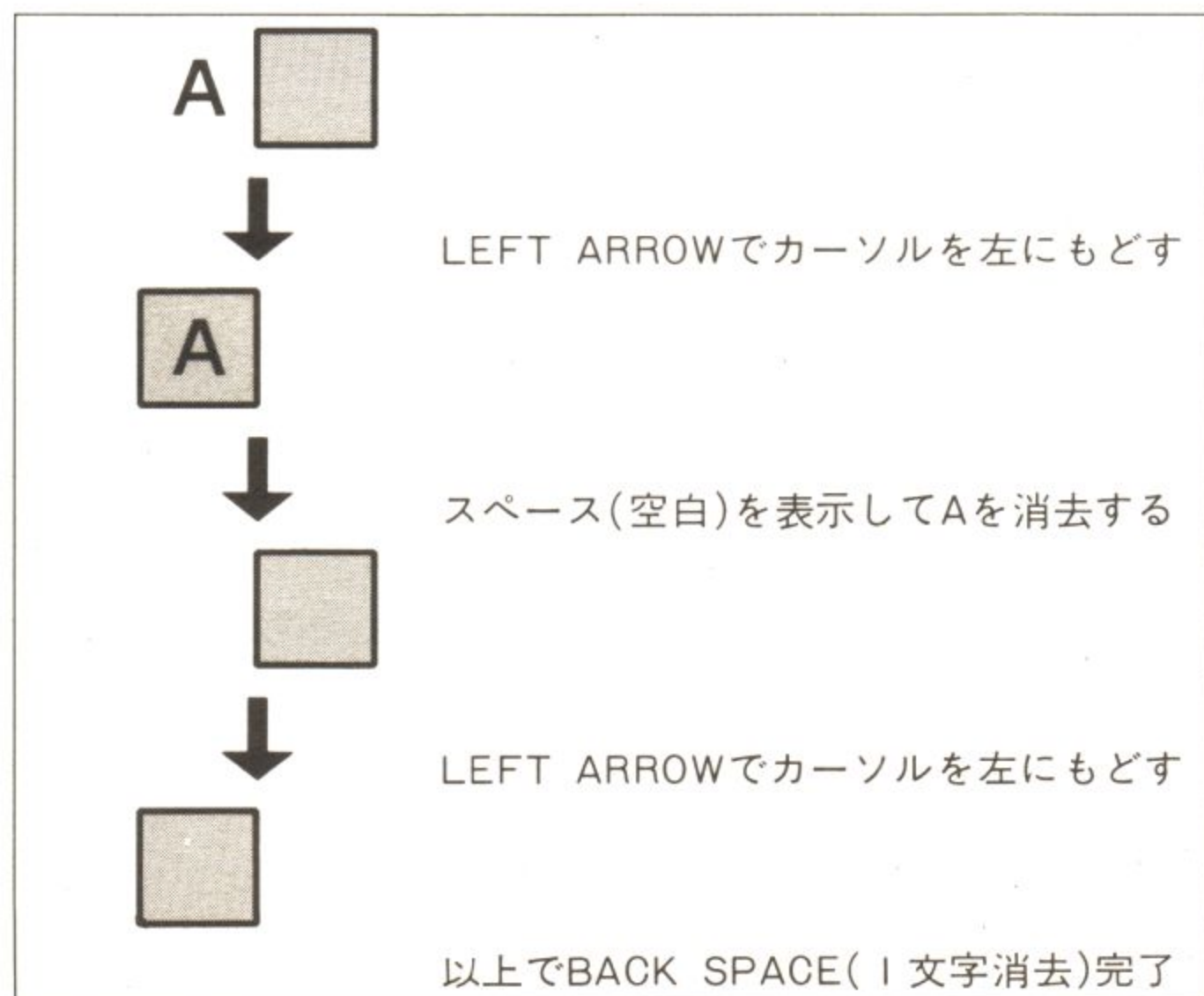
紹介した、メモリサーチ・プログラムはPC-8801用機械語開発応用ツールによって開発を行いました。

つまり、PC-8801（N88-BASIC）用のサーチプログラムをつくっておいてからPC-8001（N-BASIC）用に移植を行ったわけです。もっとも変更点は、各BASICのROM内をコールしている2個所のアドレスのみでしたから、アセンブラのシンボル定義擬似命令（EQU）を2箇所変更しただけです。

以上の点からも、前述した様に他機種への移植性を重視して組んだサーチメモリ・プログラムの移植性の良さを理解していただけたと思います。

PC-8001以外の機種をお持ちの皆さん。ぜひ御自分の機種でサーチメモリ・プログラムを走らせてみてください。意外と簡単なのでおどろかれる事と思います。

〈第103図〉 バックスペースの実現



▲NEC PC-8801mkII

マイクロ・ワード・プロセッサ I

第3ブロック最後の題材は、256バイトに収まる英文ワード・プロセッサです。

ワード・プロセッサを取り上げた理由は、コマンド解析の非常に良い実例だと考えたからで、コマンド解析を伴うプログラムであれば別にワード・プロセッサである必要はありませんでした。

ワード・プロセッサという一つのシステムが、如何にしてコマンドを解釈し、如何にして実行し

て行くか。そのようなことに興味を持っていただければ幸いです。

プログラムの仕様

このマイクロ・ワード・プロセッサは、256バイトに収まる非常に小さなプログラムですが、

〈第104図〉 マイクロ・ワード・プロセッサ印字例

<<< VERY TINY WORD PROCESSOR SPECIFICATIONS >>>

| | | |
|-----------------|---|------------------------------|
| program area | : | from c100h to c1ffh |
| buffer for edit | : | from c200h to dbffh |
| operation | : | screen edit |
| line number | : | from a to z, or from A to Z |
| title message | : | word processing system here. |
| prompt message | : | here. |
| error message | : | ring the bell only |

<< COMMANDS >>

| | | |
|--------|---|----------------------------------|
| list | : | display text |
| print | : | output text to printer |
| new | : | clear text all |
| system | : | exit from word processing system |

<< ATTENTION >>

this specifications printed with this program!
have a nice day!!

必要最小限度の機能だけは持ち合わせています。

第104図に本プログラム自身によってPC-8822（プリンタ）から打ち出した、仕様書（SPECIFICATIONS）を示します。このSPECIFICATIONSに添って、マイクロ・ワード・プロセッサの仕様を追って行きましょう。

◎メモリ・ロケーション

まず最初にプログラム・エリアですが、プログラム本体はC100H～C1FFH番地に置くことにしました。とくに今回のプログラムでは、256バイトの範囲内でどれだけのものができるかに挑戦する意味を持たせたかったため、絶対に256バイトを越えないようにプログラミングしました。

テキストを格納するためのデータ・バッファは、プログラム末すなわちC200H番地から256バイト×26行分を確保しました（第105図）。必然的に、テキストの1行はエンドマークを含めて256バイト以内でなければなりません。

◎エディタ

また、エディタの方式には、大別してポインタ・エディタとスクリーン・エディタがありますが、ここではパーソナル・コンピュータの普及と共に一般的になったスクリーン・エディット方式を採用しました。スクリーン・エディタをプログラムで実現するのは、非常に面倒であるため、N-BASICのシステム・サブルーチンを活用することにしましょう。

スクリーン・エディタには行番号が必要です。行番号には10進数値を使うのが一般的ですが、数値データを扱うには手間がかかりプログラムが長くなる恐れがあるため、アルファベットのAからZまで26文字を行番号にしました。また、この行番号は英大文字でも英小文字でも良いことにします。

行番号とテキストの間には、普通スペース（空白）等をはさみますので、ここでは1個のコロンをはさむことにしました。

〈第105図〉 テキスト・バッファのアドレス

| 番 号 | テキスト・バッファ |
|-----|---------------------------|
| 行 A | C 2 0 0 H 番地～C 2 F F H 番地 |
| 行 B | C 3 0 0 H 番地～C 3 F F H 番地 |
| 行 C | C 4 0 0 H 番地～C 4 F F H 番地 |
| 行 D | C 5 0 0 H 番地～C 5 F F H 番地 |
| 行 E | C 6 0 0 H 番地～C 6 F F H 番地 |
| 行 F | C 7 0 0 H 番地～C 7 F F H 番地 |
| 行 G | C 8 0 0 H 番地～C 8 F F H 番地 |
| 行 H | C 9 0 0 H 番地～C 9 F F H 番地 |
| 行 I | C A 0 0 H 番地～C A F F H 番地 |
| 行 J | C B 0 0 H 番地～C B F F H 番地 |
| 行 K | C C 0 0 H 番地～C C F F H 番地 |
| 行 L | C D 0 0 H 番地～C D F F H 番地 |
| 行 M | C E 0 0 H 番地～C E F F H 番地 |
| 行 N | C F 0 0 H 番地～C F F F H 番地 |
| 行 O | D 0 0 0 H 番地～D 0 F F H 番地 |
| 行 P | D 1 0 0 H 番地～D 1 F F H 番地 |
| 行 Q | D 2 0 0 H 番地～D 2 F F H 番地 |
| 行 R | D 3 0 0 H 番地～D 3 F F H 番地 |
| 行 S | D 4 0 0 H 番地～D 4 F F H 番地 |
| 行 T | D 5 0 0 H 番地～D 5 F F H 番地 |
| 行 U | D 6 0 0 H 番地～D 6 F F H 番地 |
| 行 V | D 7 0 0 H 番地～D 7 F F H 番地 |
| 行 W | D 8 0 0 H 番地～D 8 F F H 番地 |
| 行 X | D 9 0 0 H 番地～D 9 F F H 番地 |
| 行 Y | D A 0 0 H 番地～D A F F H 番地 |
| 行 Z | D B 0 0 H 番地～D B F F H 番地 |

◎メッセージ

第一のメッセージは、プログラムの起動時に一度だけ表示されるタイトル・メッセージです。

ここでは、マイクロ・ワード・プロセッサが起動したことを示すために、

word processing system here.

と、たいそうな自己主張をさせています。

第二のメッセージは、入力をうながすためのプロンプト・メッセージすなわち、

here.

です。このプロンプト・メッセージは、N-BASICでは「Ok」が表示されるタイミングと同様のタイミングで表示させています。

最後のメッセージは、入力ミスが発生したとき、すなわちコマンド（オーダー）でもテキストでもないものが入力された場合に出力する、エラー・メッセージです。

〈第106図〉 マイクロ・ワード・プロセッサ (ダンプリスト)

| | | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|
| C100 | 21 | C2 | C1 | 18 | 06 | 3E | 07 | DF | 21 | D9 | C1 | CD | BB | C1 | CD | 7E | !.....>...!.....~ |
| C110 | 1B | 38 | FB | 23 | 7E | CB | AF | FE | 41 | 38 | EA | FE | 5B | 30 | E6 | 21 | .8.#~...A8..C0.! |
| C120 | E2 | C1 | 06 | 04 | 11 | 96 | EC | 1A | CB | AF | BE | 20 | 07 | A7 | 28 | 28 |<< |
| C130 | 23 | 13 | 18 | F3 | 34 | 35 | 28 | 03 | 23 | 18 | F9 | 23 | 23 | 23 | 10 | E4 | #...45(.#..###.. |
| C140 | 11 | 96 | EC | 1A | 13 | 08 | 1A | 13 | FE | 3A | 20 | B9 | 08 | CD | B1 | C1 |:..... |
| C150 | EB | 01 | FF | 00 | ED | B0 | 18 | B6 | 23 | 5E | 23 | 56 | EB | E9 | 21 | 00 |#^#V..!. |
| C160 | C2 | AF | 06 | 1A | 77 | 24 | 10 | FC | 18 | 9E | 3E | 01 | 32 | 49 | EB | 3E |w\$.>..2I.> |
| C170 | 61 | CD | A2 | C1 | 47 | CD | F1 | 0C | 38 | 8E | 78 | 3C | FE | 7B | 20 | F1 | a...G...8.x<.<. |
| C180 | AF | 32 | 49 | EB | 18 | 82 | 3E | 61 | CD | 9C | C1 | 47 | CD | F1 | 0C | DA | .2I...>a...G.... |
| C190 | 08 | C1 | 78 | 3C | FE | 7B | 20 | F0 | C3 | 08 | C1 | C9 | DF | F5 | 3E | 3A | ..x<.<.....>: |
| C1A0 | DF | F1 | F5 | CD | B1 | C1 | CD | BB | C1 | 21 | DF | C1 | CD | BB | C1 | F1 |!..... |
| C1B0 | C9 | CB | AF | D6 | 41 | 21 | 00 | C2 | 84 | 67 | C9 | 7E | A7 | C8 | DF | 23 |A!...g.~...# |
| C1C0 | 18 | F9 | 77 | 6F | 72 | 64 | 20 | 70 | 72 | 6F | 63 | 65 | 73 | 73 | 69 | 6E | ..word processin |
| C1D0 | 67 | 20 | 73 | 79 | 73 | 74 | 65 | 6D | 20 | 68 | 65 | 72 | 65 | 2E | FC | 0D | g system here... |
| C1E0 | 0A | 00 | 4C | 49 | 53 | 54 | 00 | 86 | C1 | 50 | 52 | 49 | 4E | 54 | 00 | 6A | ..LIST...PRINT.j |
| C1F0 | C1 | 4E | 45 | 57 | 00 | 5E | C1 | 53 | 59 | 53 | 54 | 45 | 4D | 00 | 9B | C1 | .NEW.^..SYSTEM... |

本プログラムでは、CRT画面にベル・コード (07H) を出力することによって、一定時間内蔵ブザーを鳴動させています。

◎オーダー (コマンド)

マイクロ・ワード・プロセッサは、つぎに示す4種のコマンドを使用することができます。

1. LIST (テキスト表示)
2. PRINT (テキスト印字)
3. NEW (全テキスト消去)
4. SYSTEM (作業終了)

ただし、これらマイクロ・ワード・プロセッサのコマンドをN-BASIC等のコマンドやステートメントと区別するため、本文中の説明ではコマンドと呼ばずにオーダーと呼んでいます。コマンドもオーダーも日本語に訳せば、同じ命令の意味を持ちます。

なお、オーダーは英大文字、英小文字の区別なく受け付けます。

プログラムの実行方法

プログラムを実際に打ち込んで実行したい方のために、マイクロ・ワード・プロセッサの実行用ダンプリストを示します (第106図)。C100H番地からC1FFH番地までを入力ミスのないように打ち込んでください。プログラムはサブルーチン形式でリターン命令 (RET) で終結しているため、実行に際しては、N-BASICのUSR関数を利用します。

実際には、次のように操作すれば良いでしょう。

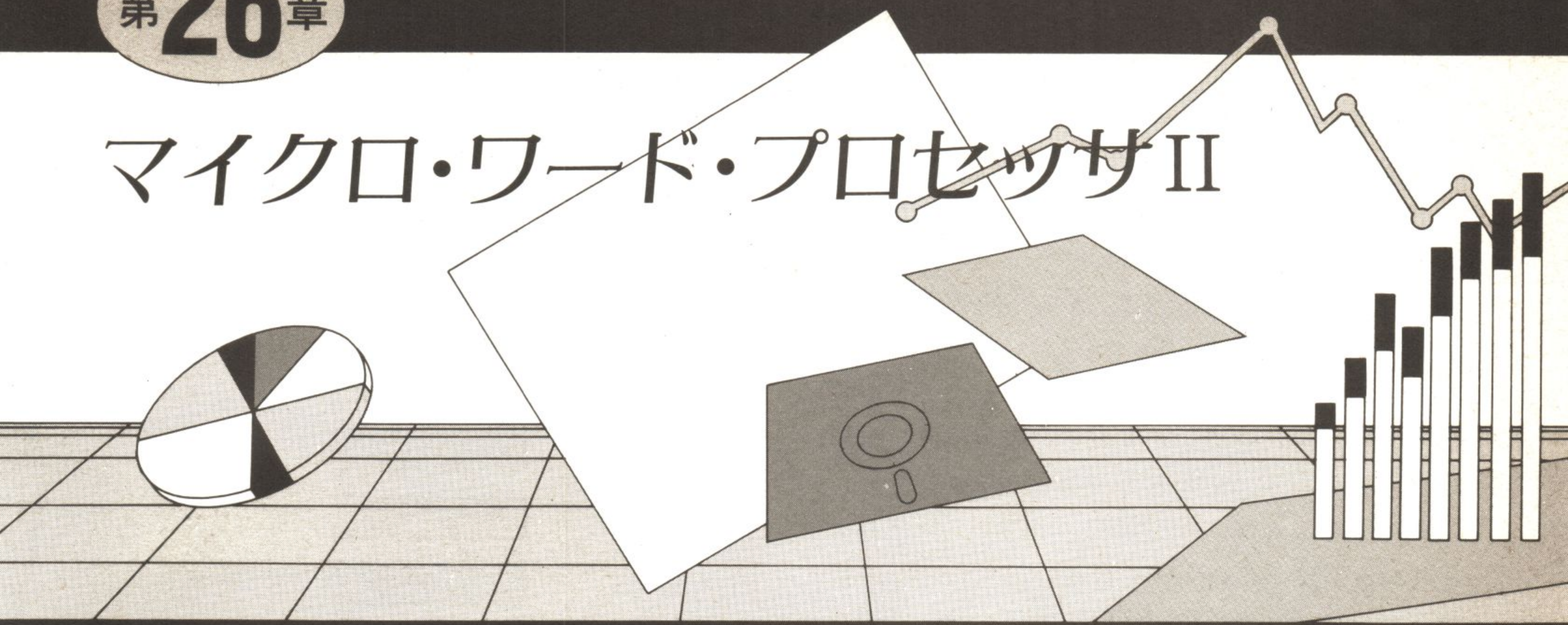
```
DEF USR=&HC100
A=USR (A)
```

また、プログラムを実行してもテキストは消去されないため、テキスト・バッファ (C200H~DBFFH番地) 上に残っている不特定なデータがテキストとして扱われてしまいます。したがって、最初にNEWオーダーを実行し、手動による全テキストの消去を行うことをお勧めします。



◀新入力方式の日本語
ワードプロセッサ
「PCWORD-M」

マイクロ・ワード・プロセッサII



プログラムの説明

前章の仕様に添ってつくったマイクロ・ワード・プロセッサのアセンブル・リスト第108図を追って行きましょう。

なお、アセンブル・リストの説明は、便宜上AブロックからMブロックまで、13のブロックに分割して行います。

◎Aブロック

本プログラムでは、3種類のN-BASICシステム・サブルーチンすなわち「OUTCHR」「CHKSTP」「SCREDT」を使用しており、2箇所のシステム・ワーク・エリア「OUTFLG」「INBUFF」を利用しています。

「OUTCHR」のラベルを付けたシステムサブルーチンは、0018番地からのサブルーチンで、Aレジスタのキャラクタ・コードに基づいたキャラクタを各周辺機器に出力するためのサブルーチンです。

この「OUTCHR」は、通常はCRT画面へ任意のキャラクタを表示するための1文字表示サブルーチンとして利用しますが、ここではワークエリア中のEB49番地「OUTFLG」の内容を01Hに変更することによって、プリンタへの1文字出力サブルーチンとしても使用しています。プリンタへの出力を終えた時点で、EB49H番地「O

UTFLG」の内容は、再び00Hにもどしておきます。

また、このサブルーチンのもう一つの特徴は、サブルーチン・コール命令(CALL)のかわりにリスタート命令(RST)を用いて呼び出すことができる点です。つまり、コール命令を用いて、

CALL 0018H

とするかわりにリスタート命令を用いて、

RST 18H

とすることができるようになります。

(実際には18Hのかわりにラベル「OUTCHR」を使用しています)

前者(コール命令)は3バイト命令、後者(リスタート命令)は1バイト命令ですから、この手法は省メモリのためにも役立っています。

また、ワード・プロセッサのLISTまたはPRINTオーダーによるテキストの表示、または印字を中断するためには、STOPキーを押下するように決めてありますが、このときSTOPキーが押されているか否かを調べる(スキャンする)ために、0CF1H番地のシステム・サブルーチンに「CHKSTP」というラベルを付けて使用しています。

このサブルーチン「CHKSTP」では、STOPキーの入力が認められればキャリ・フラグ(CY)を1にセットしてもどき、そうでなければキャリ・フラグを0にリセットしてもどきます。

最後に紹介するシステム・サブルーチンは、本

プログラムをつくる上で最も重要だといっても過言ではないサブルーチン、すなわち1B7EH番地のスクリーン・エディタ「SCREDT」です。

「SCREDT」は、スクリーン・エディット方式によってキーボードから1行分のデータを入力して、EC96H番地「INBUFF」から用意されているN-BASICのインプット・バッファに格納してもどります。ただし、入力データは254バイトまでが有効で、それを越えて入力した場合には先頭からの254バイトが入力データと見なされます。

データの輸入は、RETURN(CTRL+M)キーまたはSTOP(CTRL+C)キーの入力によって終了しますが、STOP(CTRL+C)キーによる終了時にはキャリー・フラグ(CY)を1にセットします。

このサブルーチン「SCREDT」からのリターン時にはHLレジスタ対にEC95H(INBUFF-1)が与えられます。

◎Bブロック

プログラムの先頭ではタイトル

word processing system

を表示します。

通常の場合には、さらにプロンプト・メッセージ
here.

を表示してスクリーン・エディットを行いますが、ワード・プロセッサの中でエラーが発生し、「ERROR」にジャンプしてきた場合には、コード07Hを出力することによって内蔵ブザーを鳴らしてからプロンプト・メッセージの表示に移ります。

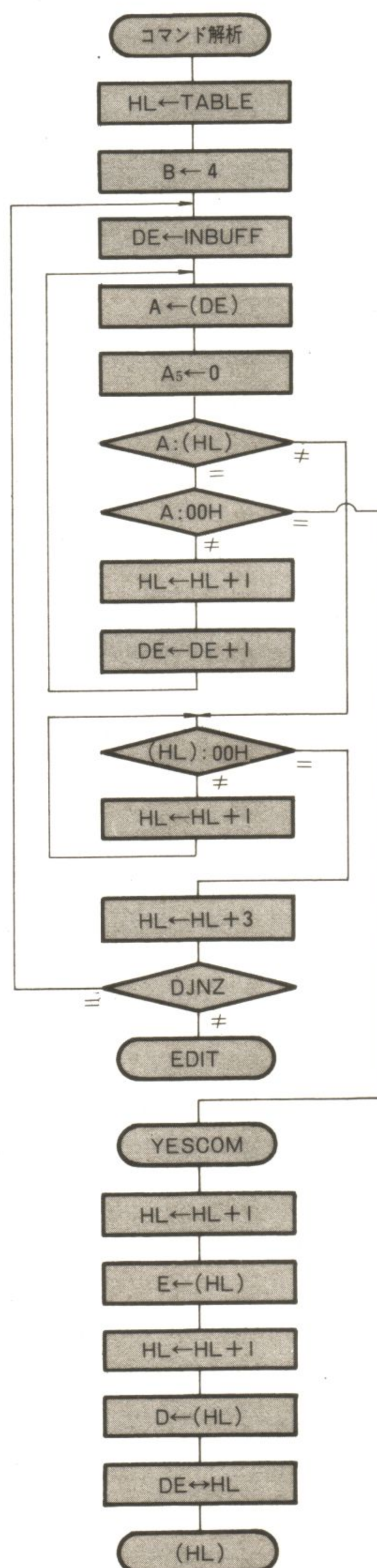
システム・サブルーチン「SCREDT」からもどった時点で最初にキャリー・フラグを調べます。ここでもしキャリー・フラグ(CY)が1にセットされている場合にはSTOPキーが押下されたわけですから、入力は無効となり、再度スクリーン・エディットを行います。

つぎに、入力データの先頭のキャラクタがアルファベットであるか否かを調べ、そうでなければ入力エラーの発生と判断してエラー処理「ERROR」にジャンプします。

さらに、入力データがオーダー（コマンド）であるか否かを判断します。このコマンド解析が本

プログラム中で最も重要でかつ複雑な部分でもありますし、文章での説明ではわかりにくいと思いますので、フローチャート（第107図）を示します。

《第107図》コマンド解析と各コマンド処理への分岐



コマンド解析により、入力データがオーダー(コマンド)であると判断した場合には「YESCOM」にジャンプして各コマンドの処理アドレスに分岐しますが、そうでない場合にはつぎの「EDIT」に入ります。

ここ「EDIT」は、スクリーン・エディタからの入力データがオーダーでない場合、すなわち行番号を伴うテキスト・データである場合の処理を一手に行う部分です。

マイクロ・ワード・プロセッサでは行番号の直後には必ずコロンを付けることになっていますから、入力データの2文字目がコロンでなければエラー発生「ERROR」です。

エラーでなければ、ブロック転送によって入力データを適切なテキスト・バッファに格納します。

◎Cブロック

Cブロック「NEW」は、ラインAからラインZまで、26行のテキストをすべて消去するための、NEWオーダーが入力された場合にジャンプしてくる処理ルーチンです。

具体的には、C200H番地から始まるテキスト・バッファの、ラインAからラインZまですべての行に対応する先頭のアドレスに、終了マークである00Hを書き込みます。

◎Dブロック

Dブロック「PRINT」は、PRINTオーダーの処理を行うためのブロックです。

最初に、「OUTFLG」に01Hをセットして出力機器をプリンタに変更し、指定行番号をAからZまで変化させながら「PRNLIN」を呼び出すことによってすべてのテキストを出力し、再び出力機器をCRTにもどします。

1行分を出力するごとに「CHKSTP」を利用してSTOPキーの押下チェックを行っており、STOPキーが押された場合には即メイン・ルーチンにもどります。

◎Eブロック

Eブロック「LIST」は、LISTオーダーの処理を行うためのブロックです。

処理自体は、出力機器をプリンタに切換えない点以外は、Dブロックと同様です。

◎Fブロック

SYSTEMオーダーが入力されると、Fブロック「SYSTEM」にジャンプしてきます。

処理自体は、リターン命令(RET)を実行することによって、ワード・プロセッサがサブルーチンとして呼び出された何らかのプログラム(通常はN-BASICインタプリタ)にもどるだけです。

◎Gブロック

このブロック「LSTLIN」は、LISTオーダー処理のEブロックから呼び出され、テキストの1行を、a～zの行番号を付けてCRT画面上に表示するためのサブルーチンです。

具体的には、Aレジスタにキャラクタ・コードで与えられたa～zの行番号をCRT画面に出力後、つづけてコロンも出力し、テキストの実際の内容を表示するため、つづくHブロックに移ります。

◎Hブロック

このブロック「PRNLIN」は、PRINTオーダー処理のDブロックから呼び出され、テキストの1行を行番号を付けずにプリンタへ出力するためのサブルーチンで、おなじ内容をCRT画面へ表示する際にも使用します。

◎Iブロック

Iブロック「CALADR」は、A～Zまたはa～zのライン番号(行番号)から、各番号に対応するテキスト・バッファの先頭アドレスを計算するためのサブルーチンです。

行番号は、AレジスタにA～Zまたはa～zのキャラクタ・コードで与えますが、サブルーチン中で英大文字に変換されるため英大文字でも英小文字でもどちらでも良く、計算したアドレスはHLレジスタ対に格納してメイン・ルーチンへもどります。

◎Jブロック

「DSPMSG」とラベルを付けたメッセージ表示サブルーチンでは、文字列（メッセージ）をCRT画面に表示、またはプリンタに印字します。

このサブルーチンを呼び出す場合には、HLレジスタ対にメッセージ・データの格納してある先頭アドレスが入っていない必要があります。

「DSPMSG」では、このHLレジスタ対をインクリメントしながらメモリ上のデータをAレジスタに取り込み、N-BASIC内のシステム・サブルーチン「OUTCHR」を利用して文字列の出力を行います。したがって、出力するデバイス（周辺機器）がCRTであるかプリンタであるかは、メイン・ルーチンからEB49H番地「OUTFLG」の内容を変更することで指定することができます。

メモリ上のデータが00H（エンド・マーク）であればメイン・ルーチンへもどります。

◎Kブロック

このプログラムでは3種類のメッセージをデータとして用意しています。

第一は、プログラムの起動時に一度だけ表示する
word processing system

のタイトル・メッセージ「TITLE」ですが、このメッセージの末尾にはエンド・マーク（00H）を付加していないため、タイトルを表示する場合にはつぎのプロンプト・メッセージも続けて表示されます。

第二は、オーダーまたはテキストの入力を促すためのプロンプト・メッセージすなわち、
here.

の末尾に、コードFCHを加え、さらに改行を行うための0DH（キャリッジ・リターン・コード）および0AH（ライン・フィード・コード）、そして00H（エンド・マーク）を加えたメッセージ・データ「PROMPT」です。

ここで、プロンプト・メッセージの直後に置かれるFCHのコードはN-BASIC独特のもので、

スクリーン・エディットのシステム・サブルーチ

ンが有している、STOP（CTRL+C）キーまたはRETURN（CTRL+M）キーによってカーソルが次行に移るような場合、その行中に表示されているメッセージの末尾にFCHが付加されていれば、スクリーン・エディットの障害にならないように自動的に消去される機能を利用するために、FCHを加えてあります。

通常、このFCHのコードを画面上などで見ることはできませんが、HAL研究所製のPCG-8100などを御利用の皆さんは、N-BASICのプロンプト・メッセージ「Ok」の直後に不特定パターンが表示されていることを御存じだと思います。実はあの不特定パターンがFCHのコードなのです。

最後のメッセージは、改行を行うための0DH（キャリッジ・リターン・コード）と0AH（ライン・フィード・コード）に00H（エンド・マーク）を加えた「CRLF」です。

しかし、このプログラムではプロンプト・メッセージ「PROMPT」の末尾に同様の改行用データを伴っているため、一つの改行用データを両方で共有しています。

したがって、実際には、用意した一連のメッセージ・データを途中から使用することによって3種類のメッセージに使いわけていることがわかります。

◎Lブロック

ワード・プロセッサの用意する4種のオーダー、すなわちLIST, PRINT, NEW, SYSTEMのキーワード自身、およびそれぞれのキーワードに対する実際の処理アドレスを保存するデータ・テーブルです。

◎Mブロック

プログラム末のC200H番地からDBFFH番地までは、ワード・プロセッサのテキストを格納するためのデータ・バッファとして使用します。

ラインAからラインZの各行に対応するテキスト・バッファのアドレスに関しては、前章第105図を御参照ください。

まとめ:あとかきにかえて

本章をもって、Z80のすべての命令の紹介を終了いたしました。

私にとっても非常に長い道程でしたが、読者の皆さんにとってはそれ以上に長い道程であったであらうと思います。御愛読ありがとうございました。

さて、マシン語に限らず、コンピュータを学ぶ上で最も重要なことは「いかにコーディングを行

うか」ではなくて「いかにアルゴリズムを組み立てるか」であり、まして「いかに多くの命令を覚えるのか」では絶対にありません。

そして、このアルゴリズムを組み立てる能力と組み立てたアルゴリズムを命令の組み合わせに置き換える能力は、実際にプログラミングを手がけることによってのみ向上すると私は思います。

皆さんが、本書で学んだ基礎知識やプログラム例を軸として、より高度な段階に進まれることを願って止みません。

《第108図》 マイクロ・ワード・プロセッサ (アセンブルリスト)

```

; PC-8001 VERY TINY WORD PROCESSOR
; Copyright 1982 (C) by KIYOSHI KAWAMURA
; in FORESIGHT PLANNING SECTION
;
;
; Aブロック
0018 OUTCHR:EQU 0018H
0CF1 CHKSTP:EQU 0CF1H
1B7E SCREDIT:EQU 1B7EH
EB49 OUTFLG:EQU 0EB49H
EC96 INBUFF:EQU 0EC96H
;
; Bブロック
; ORG 0C100H
;
C100 21C2C1 LD HL,TITLE
C103 1806 JR DSPLAY
;
C105 3E07 ERROR: LD A,07H
C107 DF RST OUTCHR
;
C108 21D9C1 START0: LD HL,PROMPT
C10B CDBBC1 DSPLAY: CALL DSPMSG
;
C10E CD7E1B START: CALL SCREDIT ; screen editor
C111 38FB JR C,START
C113 23 INC HL
C114 7E LD A,(HL)
C115 CBAF RES 5,A ; convert into capital letter
C117 FE41 CP 'A'
C119 38EA JR C,ERROR
C11B FE5B CP 'Z'+1
C11D 30E6 JR NC,ERROR
;
C11F 21E2C1 LD HL,TABLE ; top address of command table
C122 0604 LD B,4 ; set number of command
C124 1196EC START1: LD DE,INBUFF ; input buffer of n-basic
C127 1A START2: LD A,(DE)
C128 CBAF RES 5,A ; convert into capital letter
C12A BE CP (HL)
C12B 2007 JR NZ,START3
C12D A7 AND A
C12E 2828 JR Z,YESCOM
C130 23 INC HL
C131 13 INC DE
C132 18F3 JR START2
;
C134 34 START3: INC (HL)
C135 35 DEC (HL)
C136 2803 JR Z,START4
C138 23 INC HL
C139 18F9 JR START3
;

```



```

C13B 23      START4: INC HL
C13C 23      INC HL
C13D 23      INC HL
C13E 10E4     DJNZ START1

;
C140 1196EC   EDIT: LD DE, INBUFF      ; input buffer of n-basic
C143 1A      LD A, (DE)
C144 13      INC DE
C145 08      EX AF, AF'
C146 1A      LD A, (DE)
C147 13      INC DE
C148 FE3A     CP ':'
C14A 20B9     JR NZ, ERROR
C14C 08      EX AF, AF'
C14D CDB1C1   CALL CALADR             ; calculate top address of buffer
C150 EB      EX DE, HL
C151 01FF00   LD BC, OFFH
C154 EDB0     LDIR
C156 18B6     JR START

;
C158 23      YESCOM: INC HL           ;
C159 5E      LD E, (HL)              ;
C15A 23      INC HL                  ; get command address
C15B 56      LD D, (HL)              ; and jump to each command
C15C EB      EX DE, HL              ;
C15D E9      JP (HL)                ;

; new command section
;
C15E 2100C2   NEW: LD HL, BUFFER
C161 AF      XOR A
C162 061A     LD B, 'z'-'a'+1
C164 77      NEW1: LD (HL), A
C165 24      INC H
C166 10FC     DJNZ NEW1
C168 189E     JR START0

; print command section
;
C16A 3E01     PRINT: LD A, 1
C16C 3249EB   LD (OUTFLG), A          ; select printer for output

;
C16F 3E61     LD A, 'a'              ; set first line number in acc.
C171 CDA2C1   PRINT1: CALL PRNLIN    ; print out a line to printer

;
C174 47      LD B, A
C175 CDF10C   CALL CHKSTP            ; check stop key
C178 388E     JR C, START0           ; return to main routine if stop
C17A 78      LD A, B

;
C17B 3C      INC A                   ; increment line number in acc.
C17C FE7B     CP 'z'+1              ; check line number end
C17E 20F1     JR NZ, PRINT1

;
C180 AF      XOR A
C181 3249EB   LD (OUTFLG), A          ; select crt for output
C184 1882     JR START0

; list command section
;
C186 3E61     LIST: LD A, 'a'        ; set first line number
C188 CD9CC1   LIST1: CALL LSTLIN     ; display a line

;
C18B 47      LD B, A
C18C CDF10C   CALL CHKSTP            ; check stop key
C18F DA08C1   JP C, START0           ; return to main routine if stop
C192 78      LD A, B

;
C193 3C      INC A                   ; increment line number in acc.
C194 FE7B     CP 'z'+1              ; check line number end
C196 20F0     JR NZ, LIST1
C198 C308C1   JP START0

```



```

《Fブロック》      ; system command section
C19B C9            SYSTEM:RET                ; return to basic
《Gブロック》      ; list one line with line number subroutine
                  ; inputs   : line number from a to z in accumulator
                  ; destroys : register h,l
C19C DF           LSTLIN:RST OUTCHR          ; output line number
C19D F5           PUSH AF
C19E 3E3A         LD A,':'
C1A0 DF           RST OUTCHR
C1A1 F1           POP AF
《Hブロック》      ; list one line without line number subroutine
                  ; inputs   : line number from a to z in accumulator
                  ; destroys : register h,l
C1A2 F5           PRNLIN:PUSH AF
C1A3 CDB1C1       CALL CALADR                ; calculate top address of buffer
C1A6 CDBBC1       CALL DSPMSG
                  ;
C1A9 21DFC1       LD HL,CRLF
C1AC CDBBC1       CALL DSPMSG
                  ;
C1AF F1           POP AF
C1B0 C9           RET
《Iブロック》      ; calculate top address of buffer subroutine
                  ; inputs   : line number from a to z in accumulator
                  ; outputs  : top address of buffer in register hl
                  ; destroys : register a,f,h,l
C1B1 CBAF         CALADR:RES 5,A
C1B3 D641         SUB 'A'
C1B5 2100C2       LD HL,BUFFER
C1B8 84           ADD A,H
C1B9 67           LD H,A
C1BA C9           RET
《Jブロック》      ; display message subroutine
                  ; inputs   : top address of message in register hl
                  ; destroys : register a,f,h,l
C1BB 7E           DSPMSG:LD A,(HL)
C1BC A7           AND A
C1BD C8           RET Z
C1BE DF           RST OUTCHR                ; output a character
C1BF 23           INC HL
C1C0 18F9         JR DSPMSG
《Kブロック》      ; message area
C1C2 776F7264     TITLE: DEFB 'word processing system '
C1C6 2070726F
C1CA 63657373
C1CE 696E6720
C1D2 73797374
C1D6 656D20
C1D9 68657265     PROMPT:DEFB 'here.',0FCH
C1DD 2EFC
C1DF 000A00       CRLF: DEFB 0DH,0AH,0
《Lブロック》      ; command table
C1E2 4C495354     TABLE: DEFB 'LIST',0
C1E6 00
C1E7 86C1         DEFW LIST

```



```

C1E9 5052494E      DEFB 'PRINT',0
C1ED 5400
C1EF 6AC1           DEFW PRINT
C1F1 4E455700      DEFB 'NEW',0
C1F5 5EC1           DEFW NEW
C1F7 53595354      DEFB 'SYSTEM',0
C1FB 454D00
C1FE 9BC1           DEFW SYSTEM
《Mブロック》      ;
                   ; buffer for edit
                   ;
C200               BUFFER:END
    
```

ゲームマシンから

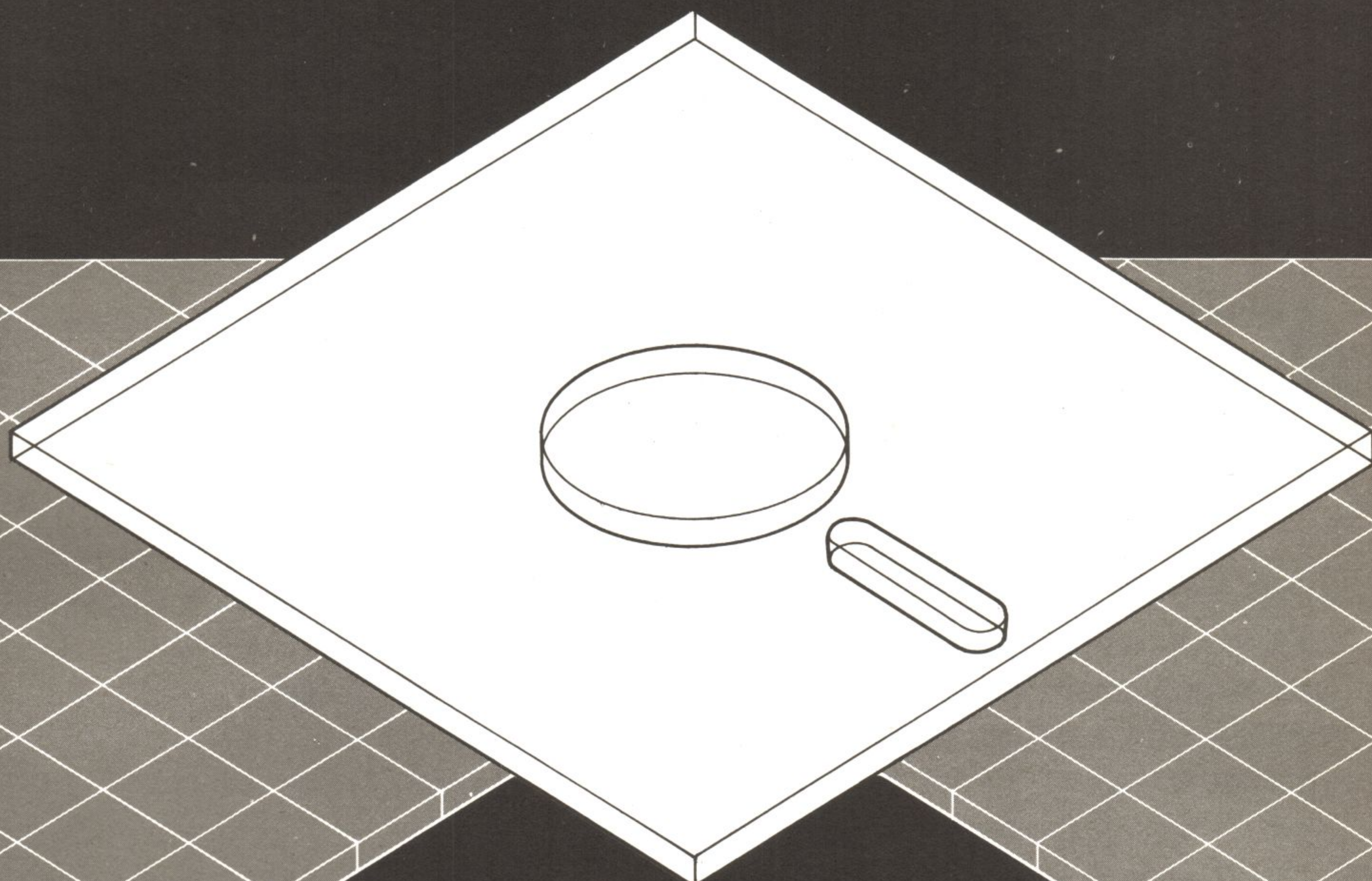
ワードプロセッサまで

期待される多機能パソコン



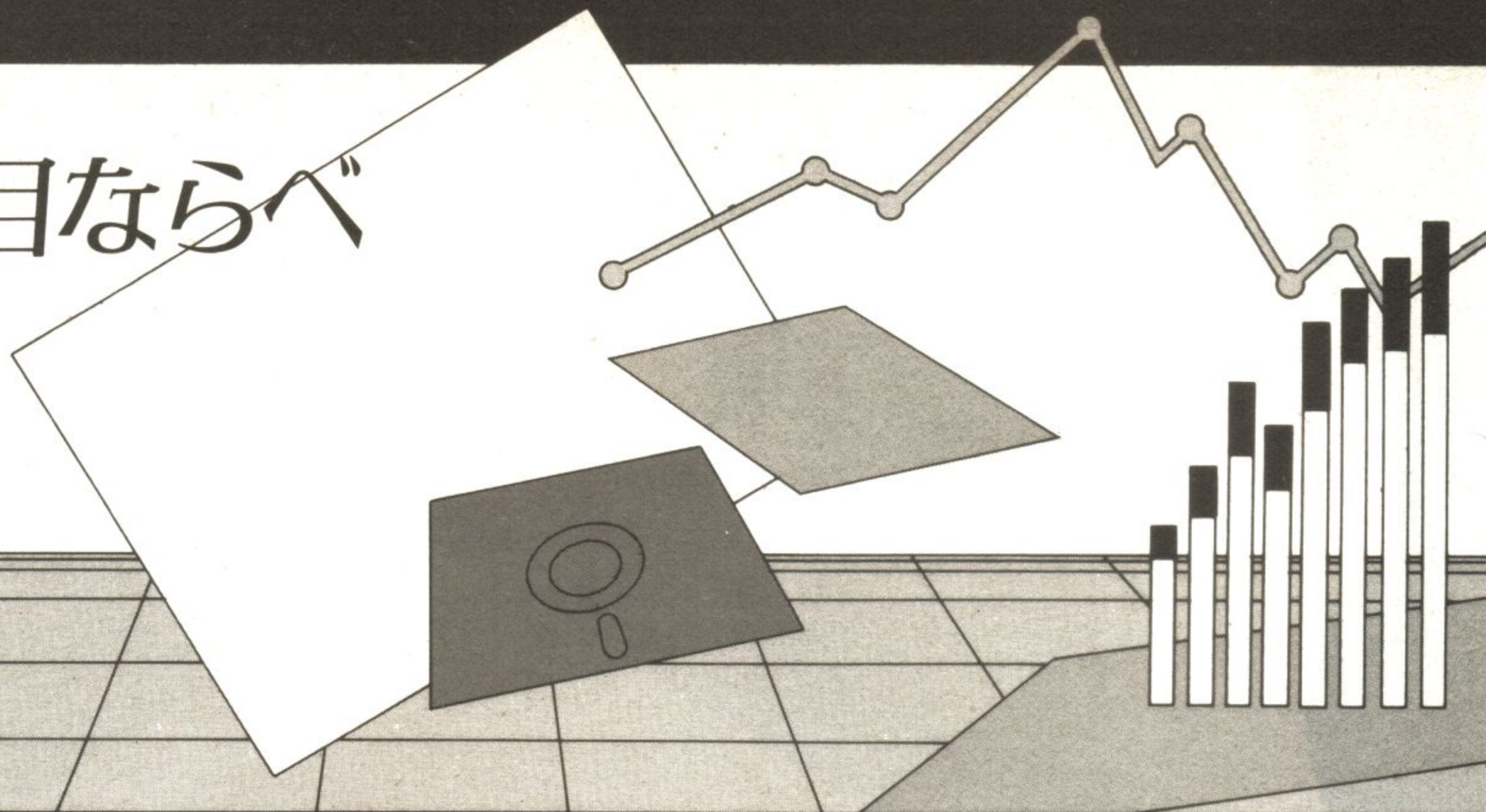
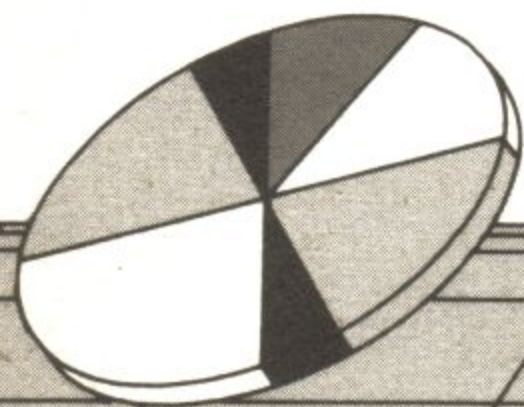
4

第 ブロック



マシン語による2次元4目ならべ

2次元4目ならべ



2次元4目ならべの概要

この2次元4目ならべゲームは、 6×6 の盤上で行う4目ならべです。

ただし、ゲーム盤は一般的に使用されるように水平に置かれた平面ではありません。幅と高さからなる壁のように直立した平面であるため、石は任意の位置に打てるわけではなく、必ず下から順番に重ね上げて行かなければなりません。

5個以上の石が並んだり、同時に2方向以上並んだ場合でも勝ちとし、36個の石すべてを打ち終わっても勝負がつかなかった場合には後手勝ちになります。

2次元4目ならべの実行方法

プログラムは、C100H番地からD5B3H番地までのメモリ上に割り当ててありますので、マシン語モニタなどを用い、実行用のダンプ・リスト（第114図）すべてを間違いのないように入力してください。

入力の前に、

CLEAR 0, &HC0FF

等を打ち込んでマシン語のための領域を確保しておけば、さらに万全といえましょう。

入力が終了したところで、かならずプログラム

をセーブしておきましょう。特にこの2次元4目ならべは5Kバイト以上の長大なプログラムですので、入力ミスが全くないということは考えられません。暴走して苦労して打ち込んだプログラムが消えてしまったら、もう2度と入力し直す気にはならないでしょう。

テープにセーブする場合には、マシン語モニタのWコマンドを用いて、

WC100, D5B3 C_R

等を行います。

また、プログラムを実行する場合には、マシン語モニタから、

GC100 C_R

を打ち込みます

2次元4目ならべの遊び方

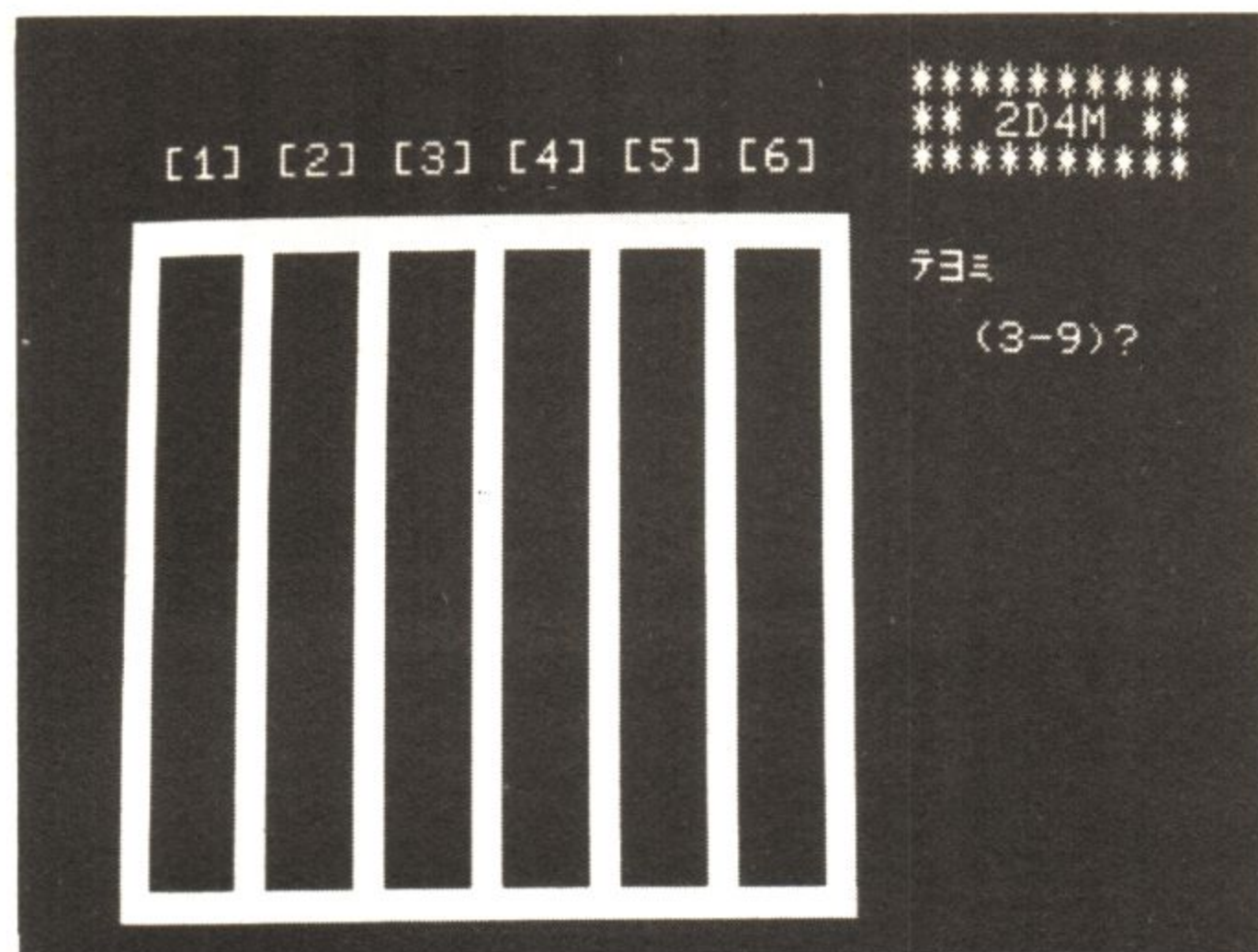
◎手読みの入力

プログラムをスタートさせると、 6×6 のゲーム盤が表示され、

テヨミ (3-9) ?

のプロンプト・メッセージと内蔵ブザーの音が入力を促します（写真1）。

ここでは、コンピュータに何手先まで読ませたいのかを3～9のキーを押下することで指定できます。すなわち、コンピュータ側の強さを3手読



《写真1》コンピュータの手読み入力

みから9手読みまでの範囲で指定することができるのですが、このプログラムでは読みの深さを増やすと処理時間が指数関数的に増大してしまいます。

この思考時間の増大に関しては、対戦相手が人間の場合には2～3分間考えていてもほとんど気にならないのですが、対戦相手がコンピュータの場合には20～30秒間が非常に待ち遠しく感じられます。したがって、気軽な気持ちで対戦を行うためには、3手読みから5手読み程度を指定するのが良いと思います。

ちなみに、私は必ず3手読みにして、お手合わせ願っております。3手読みの場合、コンピュータは0～1秒程度の思考時間で石を打ち返してきますが、それでも私は1度もコンピュータに勝ったことがありません。情けのないことですが、コンピュータの思考アルゴリズムの正しさを自ら証明しているわけですから満足している次第です。

なお、読みの深さと思考時間の関係に関しては、『コンピュータ対コンピュータ対戦成績表』（第112図～第113図）をご参照ください。

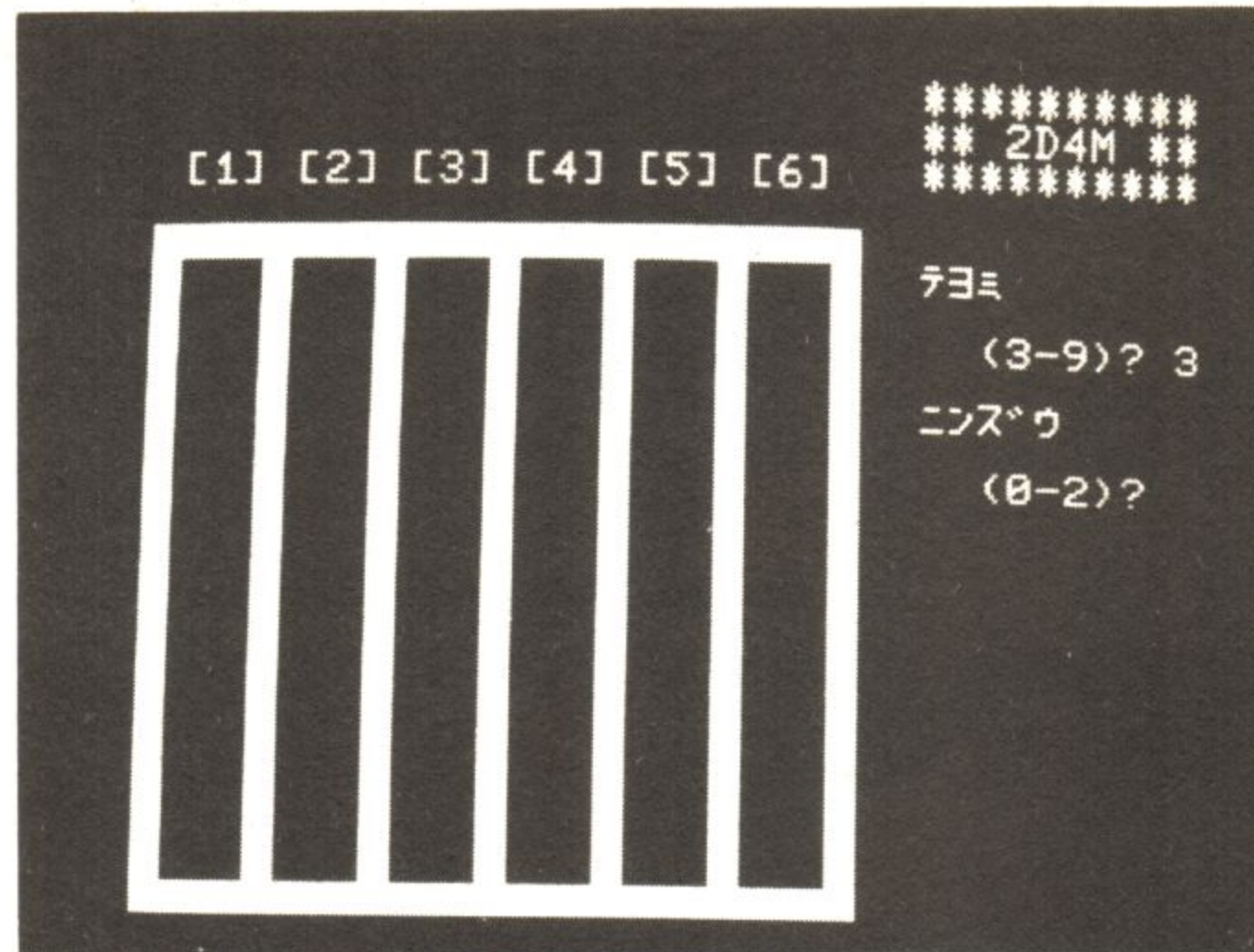
◎人数の入力

つぎに、画面は写真2のように

ニンスウ (0-2) ?

プロンプト・メッセージを表示して、プレイヤーの人数の入力を促します。

ここで、0のキーを押下した場合にはコンピュータ対コンピュータの対戦、1の場合にはコンピュータ対人間の対戦、2の場合には人間対人間の



《写真2》対戦人数の入力

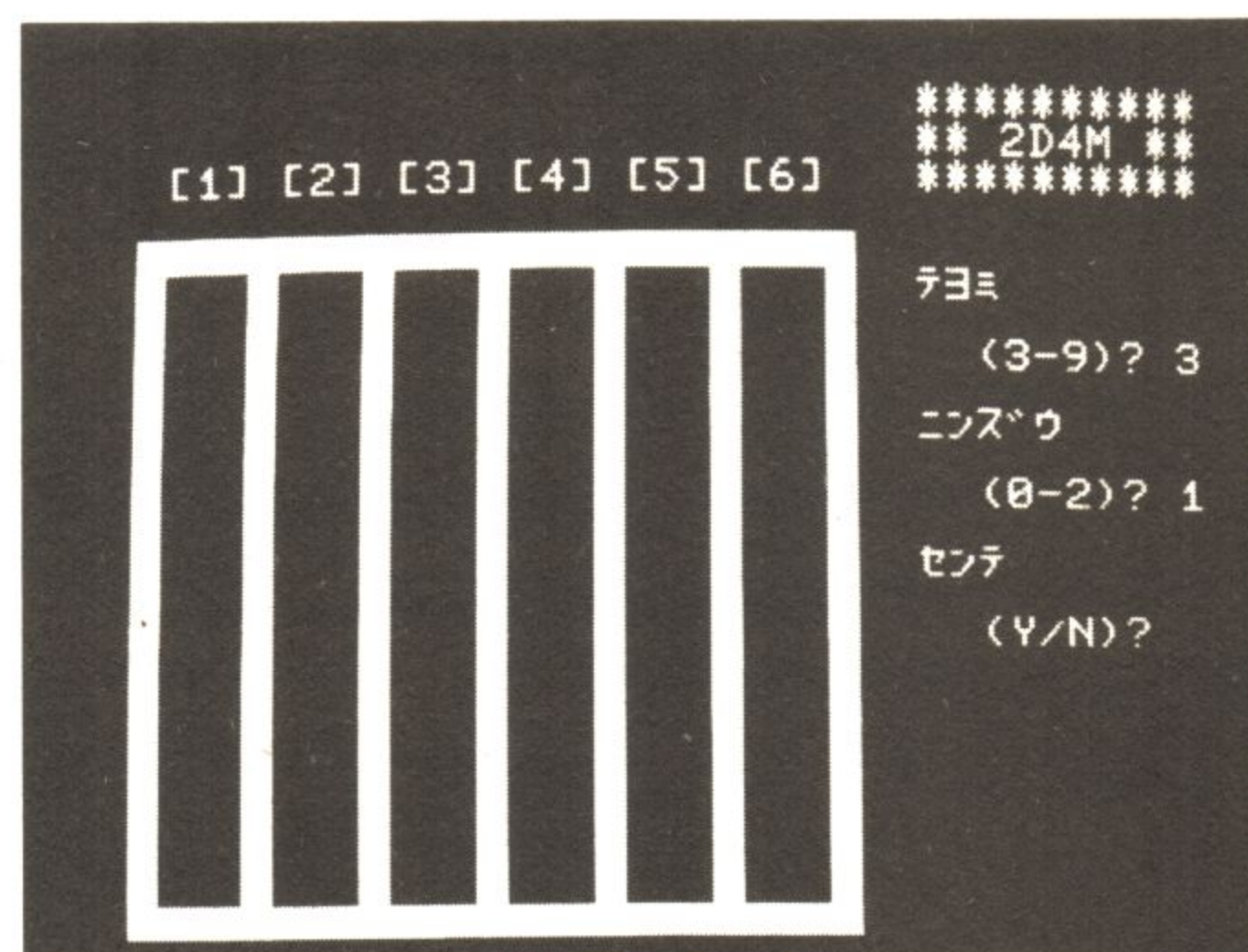
対戦となります。

◎先手／後手の入力

コンピュータ対人間の対戦を選択した場合には、さらに写真3のように、

センテ (Y/N) ?

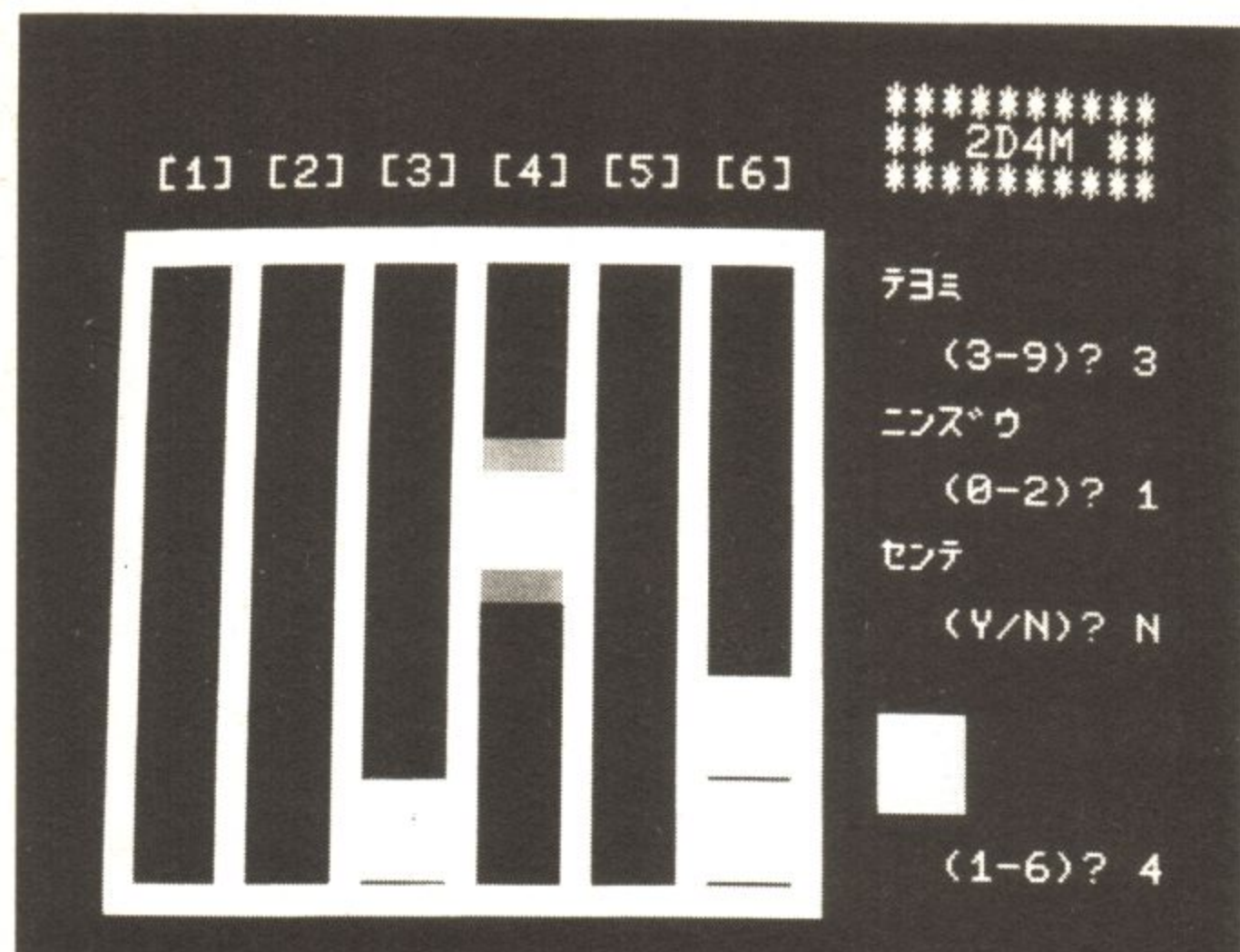
のプロンプト・メッセージを表示して、人間側が先手をとるか否かをたずねてきますので、先手をとる場合にはYのキーを、後手をとる場合にはNのキーをそれぞれ押下して答えてください。



《写真3》先手／後手の入力

◎打石

打石は1～6のキーを押下することで行います。キーを押すと、画面上部の1～6の数字の直下から石の落ちて行く様子がゆっくりとアニメーション表示されます(写真4)。



《写真4》打石の表示

非情なコンピュータは、待ったなど絶対に許してはくれません！

思う存分、戦ってください！！

◎再ゲーム

対戦が終了して再ゲームを行う場合には、RETURNキーを押下します。

また、入力待ちのときにSTOPキーを入力すれば、ゲームを中断することができます。

コンピュータの思考アルゴリズム

コンピュータの思考には、一般的なミニ・マックス (MIN-MAX) 法(第109図)を採用していますが、加えて同時に必勝手順の探索も可能としており、先読みの処理で最後に置かれた石の周囲から評価値を計算しています。

評価値は、0～255 (00H～FFH) の値をとりますが、自分の石が4個並んだ場合には200 (C8H)、相手の石が並んだ場合には1 (01H) となって、その枝の探索を打ち切り、別の枝へと探索を進めます。

この手のプログラムをつくるためには、いかにコンピュータに思考させるかというアルゴリズムを考える部分が非常に大きなウェイトを占め、組み立てたアルゴリズムを命令の組み合わせに置き換える能力 (プログラミング能力) は二の次だといっても過言ではありません。

上記のようなわけで、マシン語の入門書であるはずの本書の教材に、この2次元4目ならべを選択したことを皆さんにおわびするべきかも知れません。

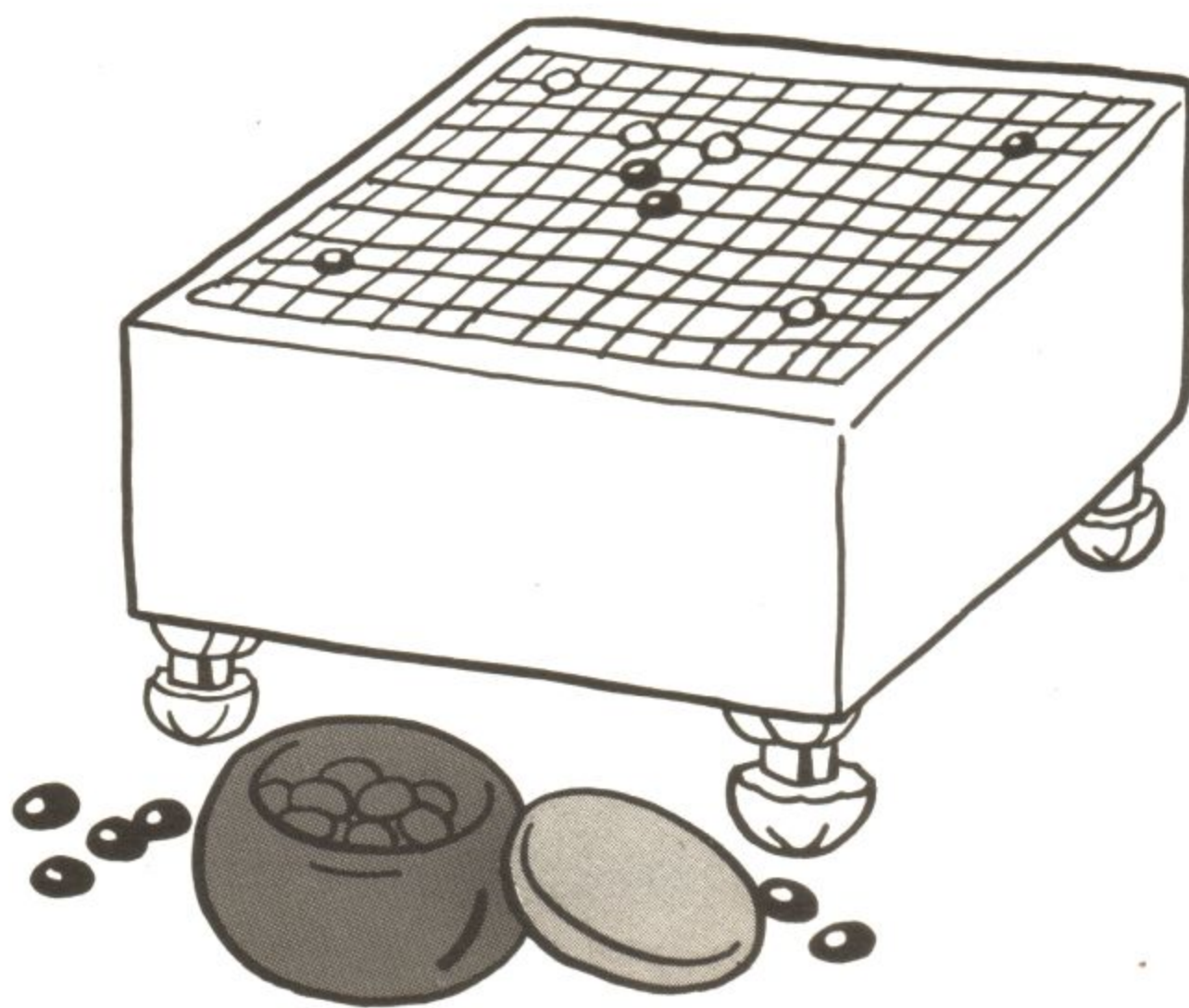
しかし、このゲームを解析することにより、マシン語の勉強と同時にプログラム作成の上で重要なアルゴリズムの組み方の参考にしていただけることと思います。また、マシン語のゲームだけやりたいという方は、それでもかまいません。このゲームで遊んでくださることにより、コンピュータの能力の一片、すなわち『思考への可能性』を感じていただけることだと確信いたします。

プログラムの開発

このプログラムは、はじめFORESIGHTの独立コンパイラによって製作しました。このためプログラムの構造やワーク・エリアの使い方等に、多数無駄な部分が存在します。

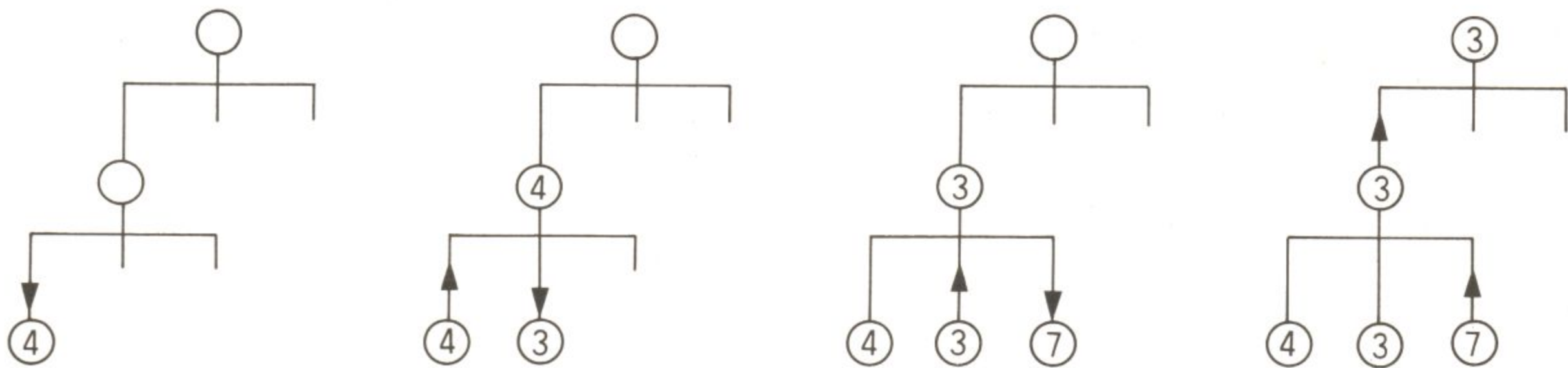
その後、より高速化、また多機能化を実現したくなったため、独立コンパイラのオブジェクト・コードから今度はアセンブラのソース・プログラムを生成し直すことになりました。

このようにして生成したアセンブラのソース・プログラムに改良をかさね、さらにアセンブラによってアセンブルし直したものが、ここに掲載させていただきました2次元4目ならべのプログラム・リストです。

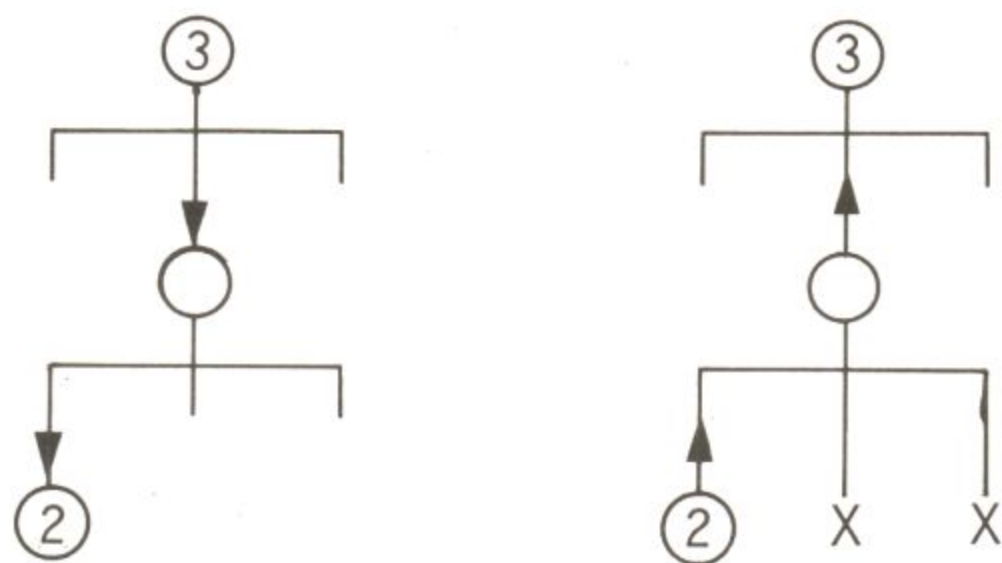


《第109図》MIN-MAX評価の原理

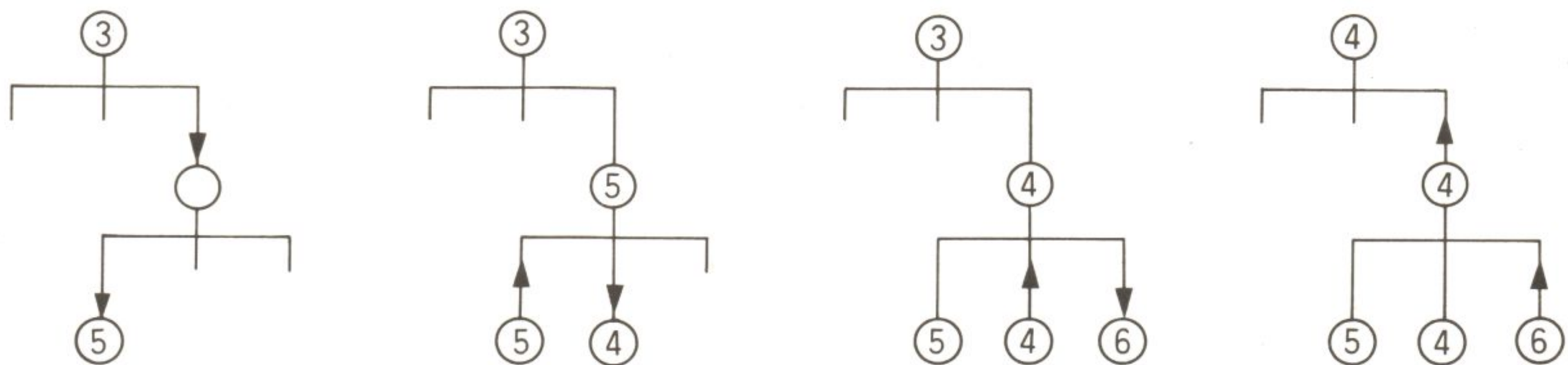
MIN-MAX評価の基本原則



● 4, 3, 7の最小値をとる。

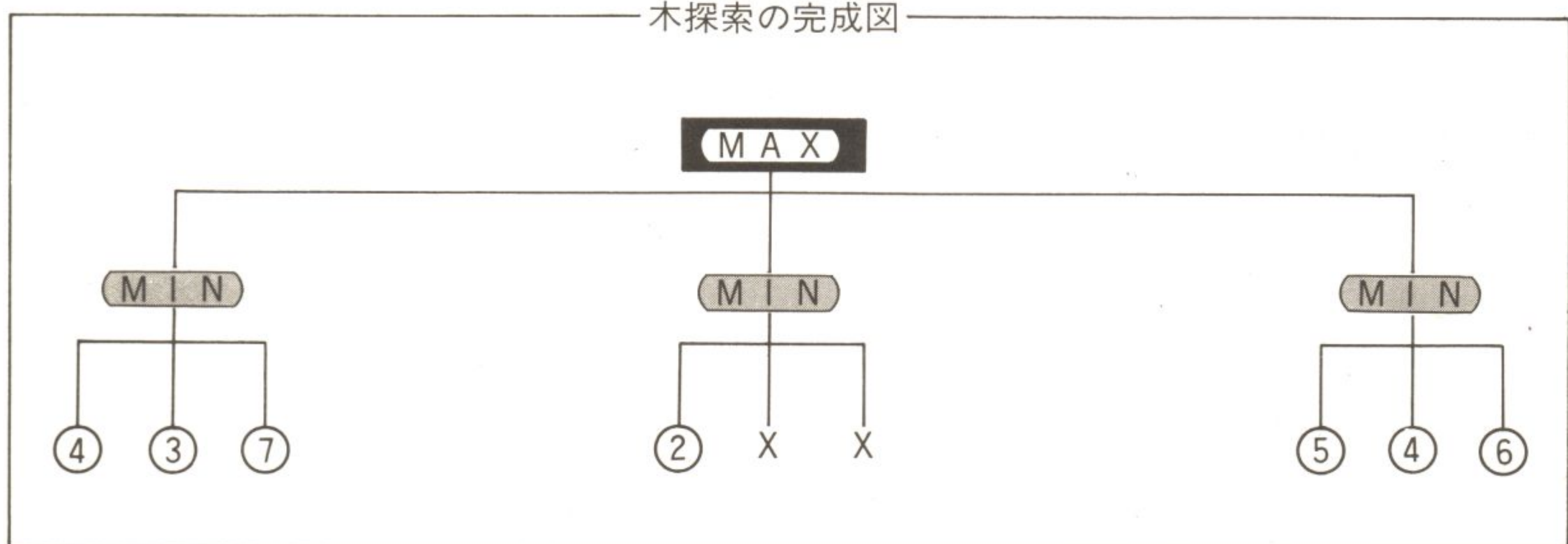


●ここで、 $2 < 3$ であるから、他の枝の評価が2未満でも2以上でもこの枝が選択される可能性はなく、 $\alpha - \beta$ 枝刈りが行われる。

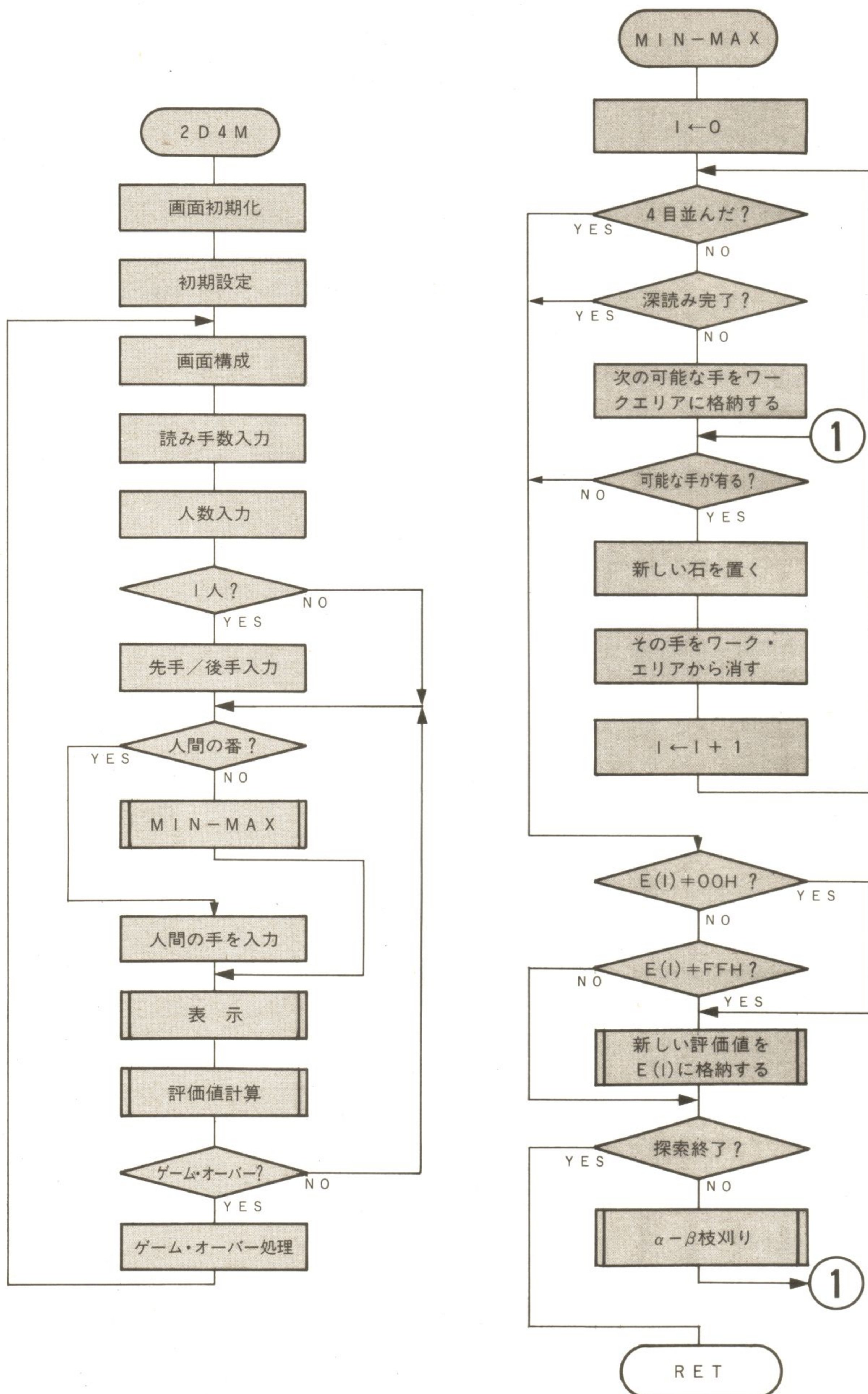


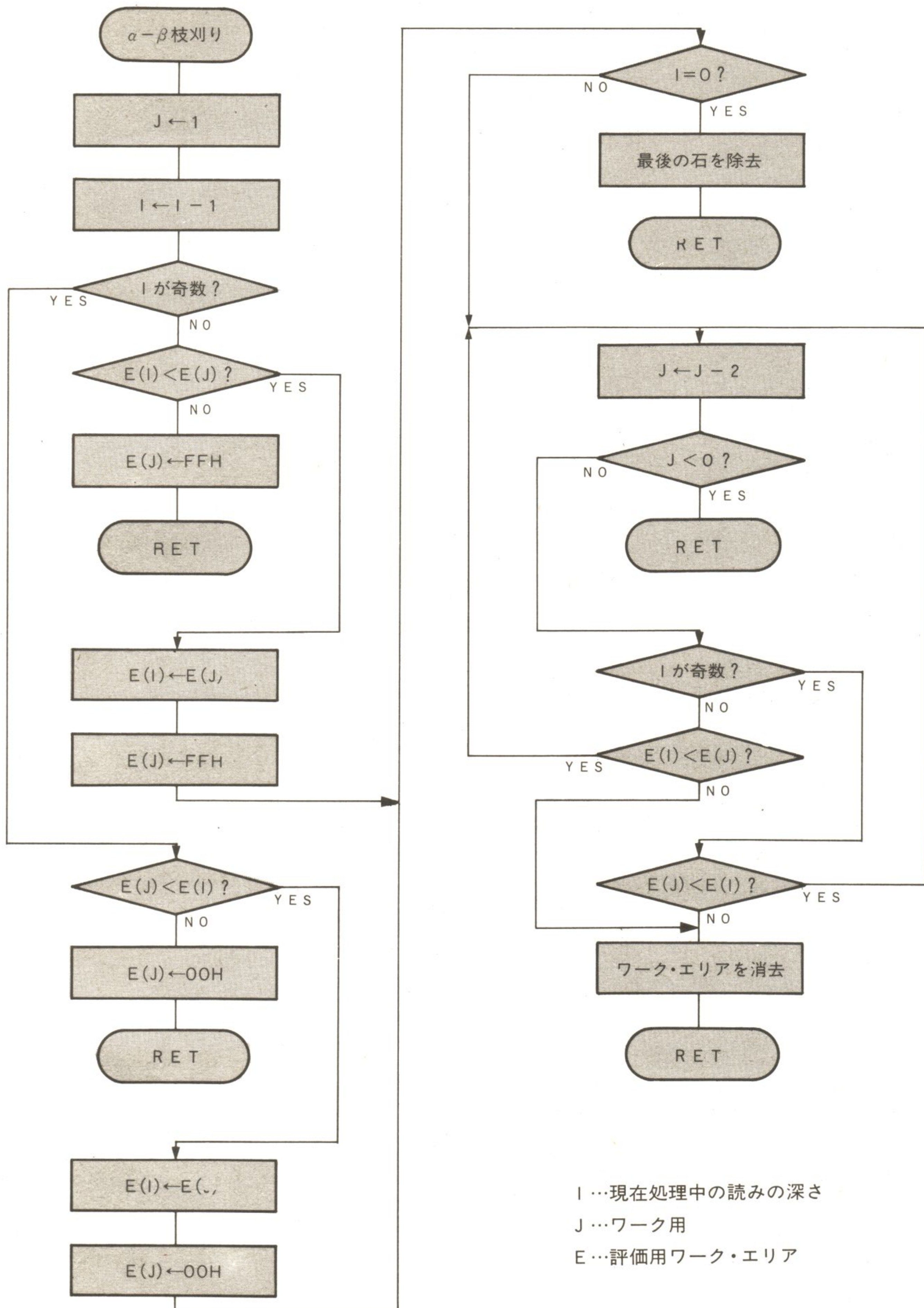
● 5, 4, 6の最少数値4と、以前の評価値3との最大値4をとる。

木探索の完成図



《第110図》 2次元4目ならべ・ゼネラルフローチャート

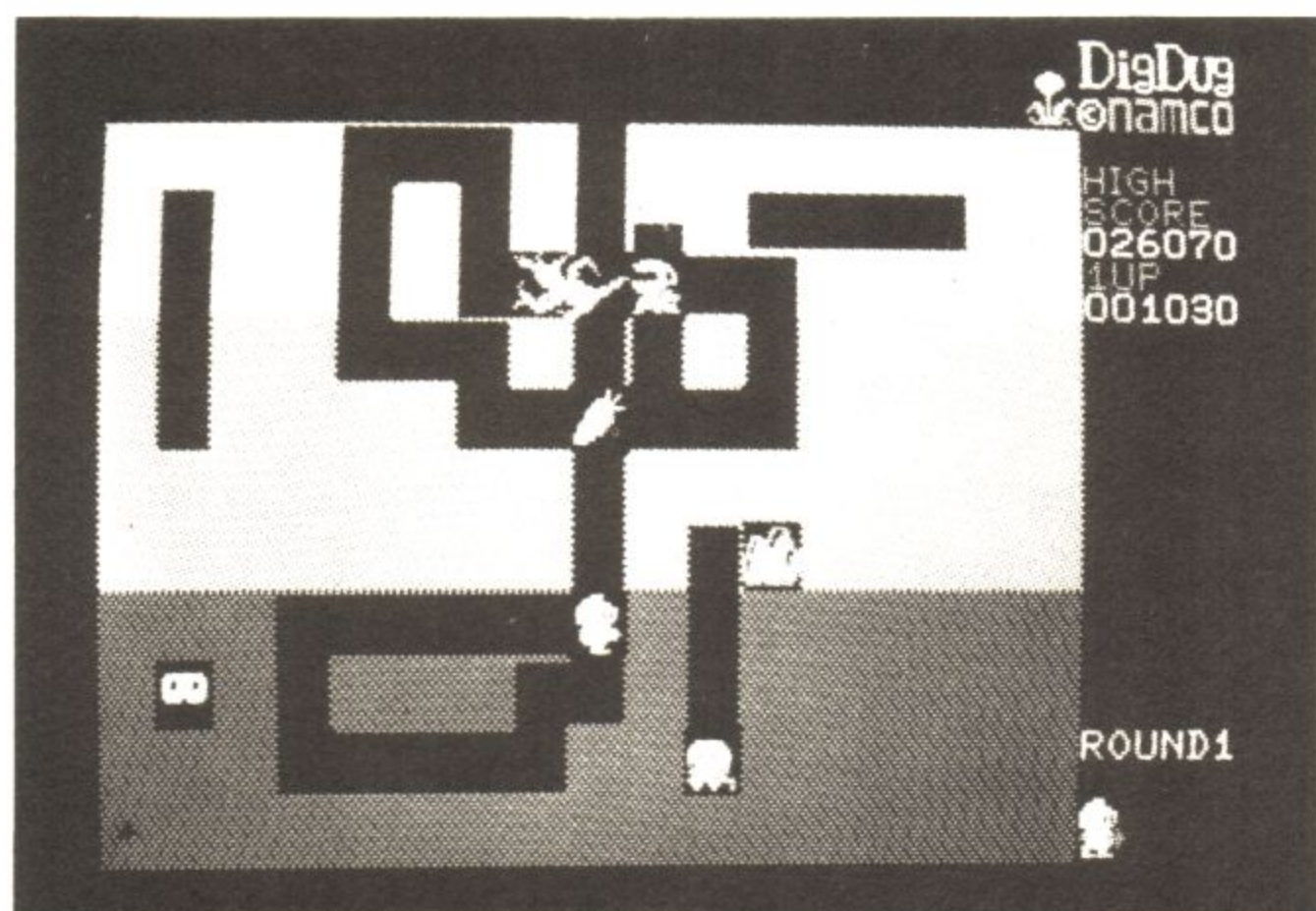




《第111図》 2次元4目ならべ・ワークエリア・マップ

| エ リ ア | 名 称 | 内 容 |
|-----------------------|----------------------------|--|
| C 4 0 0 H ~ C 4 2 3 H | | 評価用ゲーム盤。 |
| C 4 2 4 H ~ C 4 2 B H | | 探索方向指定用データ・テーブル。 |
| C 4 2 C H | | 評価バッファ。容量は読みの深さに準じて拡大する。 |
| E A 5 8 H | 『T R M F L G』 | N - B A S I Cのターミナル・モード・フラグ。サブルーチン・パッケージがM S Bをプリンタ・スイッチとして使用。 |
| E A 5 B H | 『A T R C O D』 | N - B A S I Cのアトリビュート・コード設定。 |
| E A 5 C H | | N - B A S I Cのファンクション・キー表示方法設定。 |
| E A 5 D H | 『R O L L I N』 | N - B A S I Cのスクロール・エリア開始行番号。 |
| E A 5 E H | | N - B A S I Cのスクロール・エリア行数。 |
| E A 6 3 H | 『C U S P O S』 | N - B A S I Cのカーソル縦位置。 |
| E A 6 4 H | | N - B A C I Cのカーソル横位置。 |
| E C 9 5 H ~ E C F F H | 『I N B U』 | インプット・バッファ I。 |
| E C 9 6 H ~ E C F F H | 『I N B U F F』 | インプット・バッファ II。 |
| E C 9 7 H ~ E C F F H | 『I N B U F』 | インプット・バッファ III。 |
| E D 0 0 H ~ E D 9 5 H | 『B A S E』 | 第2ワーク・エリア。 |
| E D 0 6 H ~ E D 0 7 H | 『S T O N E』 『S O T N E』 | 石の種類。1のとき先手側，2のとき後手側をしめす。 |
| E D 0 8 H ~ E D 0 9 H | 『P L A Y S』 | 手数。 |
| E D 0 A H ~ E D 0 B H | 『P L A Y E R』 | プレーヤーの人数。0のときコンピュータ対コンピュータの対戦，1のときコンピュータ対人間の対戦，2のとき人間対人間の対戦をしめす。 |
| E D 0 C H ~ E D 0 D H | 『S T O N E I』 | 『S T O N E』『S O T N E』の一時退避用。 |
| E D 0 E H ~ E D 0 F H | 『W O R K I』 | ループ制御用等。 |
| E D I 0 H ~ E D I I H | 『C O O R D I』 | 手の一時退避用。 |
| E D I 2 H ~ E D I 3 H | 『C O O R D 2』 | 手の一時退避用。 |
| E D I 4 H ~ E D I 5 H | 『W O R K 2』 | ループ制御用等。 |
| E D I 6 H ~ E D I 7 H | 『D E E P』 | 処理の深さ×8を保存。 |
| E D I 8 H ~ E D I 9 H | 『D E E P I』 | 『D E E P』の一時退避用。 |
| E D I A H ~ E D I B H | 『W O R K 3』 | ループ制御用等。 |
| E D I C H ~ E D I D H | 『W O R K 4』 | ループ制御用等。 |
| E D I E H ~ E D I F H | 『V E C T I』 | 探索の横方向。 |
| E D 2 0 H ~ E D 2 I H | 『V E C T 2』 | 探索の縦方向。 |
| E D 2 2 H ~ E D 2 3 H | 『S P A C E S』 | 盤上で石の置いていない場所の数。 |
| E D 2 4 H ~ E D 2 5 H | 『C O O R D 3』 | 『C O O R D I』の一時退避用。 |
| E D 2 6 H ~ E D 2 7 H | 『C O O R D 4』 | 『C O O R D 2』の一時退避用。 |
| E D 2 8 H ~ E D 2 9 H | 『V A L U E』 | 評価値。相手の石が4個並んだとき1を，自分の石が4個並んだとき200をとる。 |
| E D 2 A H ~ E D 2 B H | 『R D E E P』 | コンピュータが思考する読みのふかさ×8を保存。 |
| E D 2 C H ~ E D 2 D H | 『A B F L A G』 | $\alpha - \beta$ 枝刈りを実施するか否かのフラグ。0のときOFF，1のときON。 |

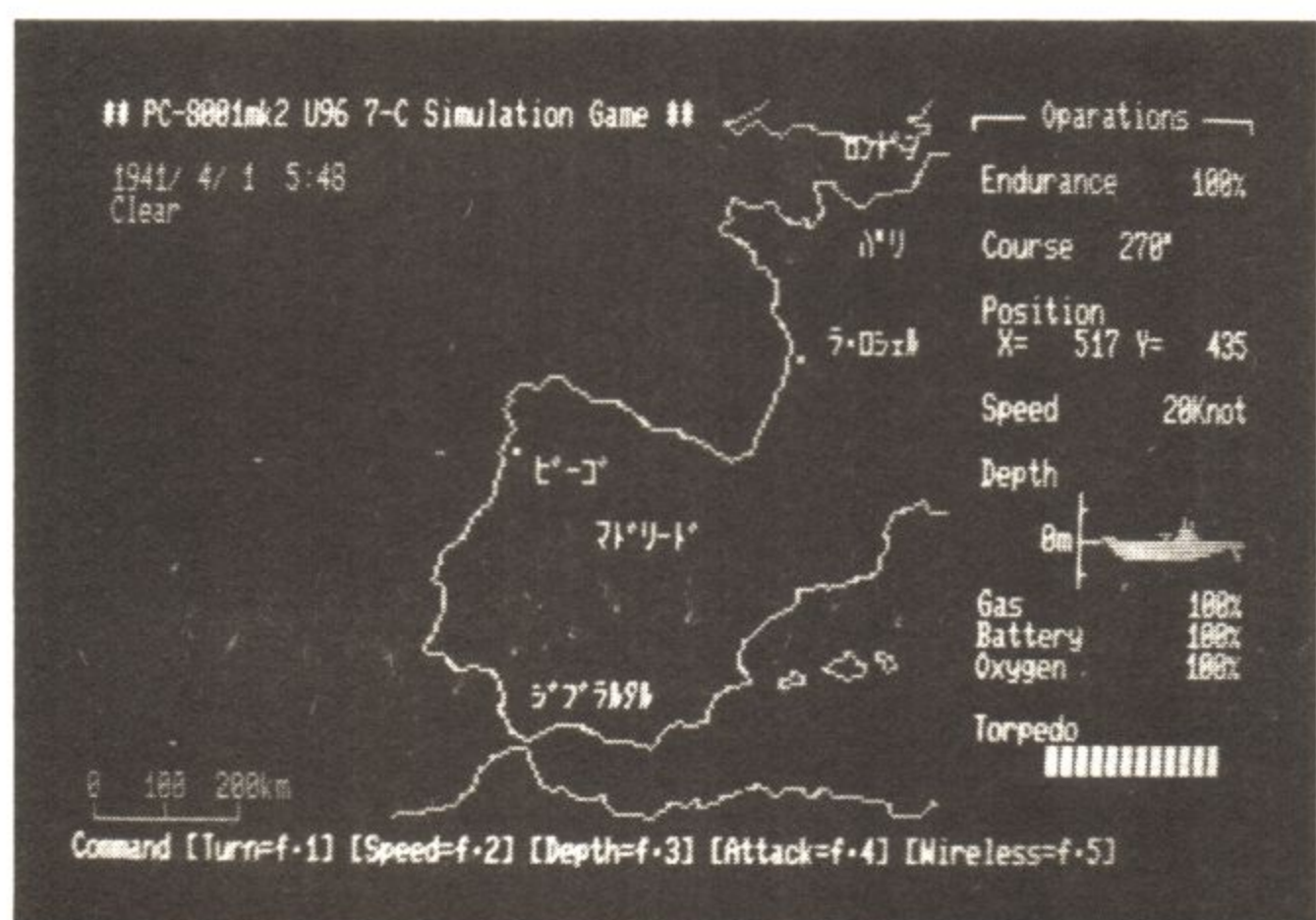
| | | |
|-------------|----------|--|
| ED2EH~ED2FH | 『PATR』 | N-BASICシステム・ワーク・エリア中のアトリビュート・コード格納アドレスを保存。 |
| ED30H~ED31H | 『VECTOR』 | 探索方向指定用データ・テーブルの先頭アドレスを保存。 |
| ED32H~ED33H | 『STACK』 | 評価スタックの先頭アドレスを保存。 |
| ED34H~ED35H | 『COORD』 | 手。 |
| ED36H~ED37H | 『PCUS』 | N-BASICシステム・ワーク・エリア中のカーソル・ポジション格納アドレスを保存。 |
| ED38H~ED39H | 『BOARD』 | 評価バッファの先頭アドレスを保存。 |
| ED4EH~ED4FH | 『DIVWK』 | 除算用。剰余を保存。 |
| ED52H~ED53H | 『RNDBUF』 | 乱数値。 |
| ED68H~ED69H | 『DIVWK』 | 除算用。 |
| ED87H~ED95H | 『UBUF』 | 数値▷文字変換バッファI。 |
| ED88H~ED95H | 『UBUFF』 | 数値▷文字変換バッファII。 |
| FF3EH~FFFFH | 『BOTTOM』 | スタック・エリア。 |



PC-8801ディグダグ



PC-8801 ZENON



PC-8001mkII U-Boat



《第112図》 2次元4目ならべ3手読みにおけるコンピュータ対コンピュータ対戦成績表

| 手数 | 手番 | 手 | 経過時間 | 思考所要時間 | 対戦結果 |
|------|----|---|------------|------------|--------|
| 1手目 | 先手 | 2 | 00時間00分00秒 | 00時間00分00秒 | |
| 2手目 | 後手 | 3 | 00時間00分00秒 | 00時間00分00秒 | |
| 3手目 | 先手 | 2 | 00時間00分00秒 | 00時間00分00秒 | |
| 4手目 | 後手 | 3 | 00時間00分01秒 | 00時間00分01秒 | |
| 5手目 | 先手 | 3 | 00時間00分02秒 | 00時間00分01秒 | |
| 6手目 | 後手 | 2 | 00時間00分04秒 | 00時間00分02秒 | |
| 7手目 | 先手 | 5 | 00時間00分05秒 | 00時間00分01秒 | |
| 8手目 | 後手 | 2 | 00時間00分06秒 | 00時間00分01秒 | |
| 9手目 | 先手 | 3 | 00時間00分07秒 | 00時間00分01秒 | |
| 10手目 | 後手 | 3 | 00時間00分08秒 | 00時間00分01秒 | |
| 11手目 | 先手 | 2 | 00時間00分09秒 | 00時間00分01秒 | |
| 12手目 | 後手 | 6 | 00時間00分10秒 | 00時間00分01秒 | |
| 13手目 | 先手 | 4 | 00時間00分11秒 | 00時間00分01秒 | |
| 14手目 | 後手 | 6 | 00時間00分11秒 | 00時間00分00秒 | |
| 15手目 | 先手 | 6 | 00時間00分12秒 | 00時間00分01秒 | |
| 16手目 | 後手 | 1 | 00時間00分13秒 | 00時間00分01秒 | |
| 17手目 | 先手 | 4 | 00時間00分14秒 | 00時間00分01秒 | |
| 18手目 | 後手 | 4 | 00時間00分15秒 | 00時間00分01秒 | |
| 19手目 | 先手 | 4 | 00時間00分16秒 | 00時間00分01秒 | |
| 20手目 | 後手 | 4 | 00時間00分17秒 | 00時間00分01秒 | |
| 21手目 | 先手 | 4 | 00時間00分18秒 | 00時間00分01秒 | |
| 22手目 | 後手 | 3 | 00時間00分18秒 | 00時間00分00秒 | |
| 23手目 | 先手 | 2 | 00時間00分19秒 | 00時間00分01秒 | |
| 24手目 | 後手 | 6 | 00時間00分19秒 | 00時間00分00秒 | |
| 25手目 | 先手 | 6 | 00時間00分20秒 | 00時間00分01秒 | |
| 26手目 | 後手 | 5 | 00時間00分20秒 | 00時間00分00秒 | |
| 27手目 | 先手 | 6 | 00時間00分20秒 | 00時間00分00秒 | |
| 28手目 | 後手 | 5 | 00時間00分21秒 | 00時間00分01秒 | |
| 29手目 | 先手 | 5 | 00時間00分21秒 | 00時間00分00秒 | |
| 30手目 | 後手 | 5 | 00時間00分21秒 | 00時間00分00秒 | |
| 31手目 | 先手 | 1 | 00時間00分21秒 | 00時間00分00秒 | |
| 32手目 | 後手 | 1 | 00時間00分22秒 | 00時間00分01秒 | |
| 33手目 | 先手 | 1 | 00時間00分22秒 | 00時間00分00秒 | |
| 34手目 | 後手 | 1 | 00時間00分22秒 | 00時間00分00秒 | |
| 35手目 | 先手 | 5 | 00時間00分23秒 | 00時間00分01秒 | |
| 36手目 | 後手 | 1 | 00時間00分23秒 | 00時間00分00秒 | 〈後手勝ち〉 |

[1] [2] [3] [4] [5] [6]

| | | | | | |
|----|----|----|----|----|----|
| 36 | 23 | 22 | 21 | 35 | 27 |
| 34 | 11 | 10 | 20 | 30 | 25 |
| 33 | 8 | 9 | 19 | 29 | 24 |
| 32 | 6 | 5 | 18 | 28 | 15 |
| 31 | 3 | 4 | 17 | 26 | 14 |
| 16 | 1 | 2 | 13 | 7 | 12 |

** 2D4M **

テヨミ

(3-9)? 3

ニンズウ

(0-2)? 0



(1-6)? 1

《第113図》2次元4目ならべ9手読みにおけるコンピュータ対コンピュータ対戦成績表

| 手数 | 手番 | 手 | 過時間 | 思考所要時間 | 対戦結果 |
|------|----|---|------------|------------|--------|
| 1手目 | 先手 | 2 | 00時間00分00秒 | 00時間00分00秒 | 〈後手勝ち〉 |
| 2手目 | 後手 | 1 | 00時間00分00秒 | 00時間00分00秒 | |
| 3手目 | 先手 | 4 | 00時間00分00秒 | 00時間00分00秒 | |
| 4手目 | 後手 | 5 | 00時間35分39秒 | 00時間35分39秒 | |
| 5手目 | 先手 | 5 | 00時間59分17秒 | 00時間23分38秒 | |
| 6手目 | 後手 | 5 | 01時間23分25秒 | 00時間24分08秒 | |
| 7手目 | 先手 | 4 | 01時間46分23秒 | 00時間22分58秒 | |
| 8手目 | 後手 | 5 | 02時間19分25秒 | 00時間33分02秒 | |
| 9手目 | 先手 | 5 | 02時間34分12秒 | 00時間14分47秒 | |
| 10手目 | 後手 | 6 | 02時間56分54秒 | 00時間22分42秒 | |
| 11手目 | 先手 | 6 | 03時間12分17秒 | 00時間15分23秒 | |
| 12手目 | 後手 | 1 | 03時間26分38秒 | 00時間14分21秒 | |
| 13手目 | 先手 | 4 | 03時間42分26秒 | 00時間15分48秒 | |
| 14手目 | 後手 | 4 | 03時間53分50秒 | 00時間11分24秒 | |
| 15手目 | 先手 | 6 | 04時間11分24秒 | 00時間17分34秒 | |
| 16手目 | 後手 | 6 | 04時間26分10秒 | 00時間14分46秒 | |
| 17手目 | 先手 | 4 | 04時間41分09秒 | 00時間14分59秒 | |
| 18手目 | 後手 | 6 | 05時間05分23秒 | 00時間24分14秒 | |
| 19手目 | 先手 | 1 | 05時間10分41秒 | 00時間05分22秒 | |
| 20手目 | 後手 | 6 | 05時間12分31秒 | 00時間01分50秒 | |
| 21手目 | 先手 | 2 | 05時間14分52秒 | 00時間02分21秒 | |
| 22手目 | 後手 | 2 | 05時間14分52秒 | 00時間00分00秒 | |
| 23手目 | 先手 | 5 | 05時間15分30秒 | 00時間00分38秒 | |
| 24手目 | 後手 | 2 | 05時間15分58秒 | 00時間00分28秒 | |
| 25手目 | 先手 | 4 | 05時間16分11秒 | 00時間00分13秒 | |
| 26手目 | 後手 | 2 | 05時間16分15秒 | 00時間00分04秒 | |
| 27手目 | 先手 | 3 | 05時間16分18秒 | 00時間00分03秒 | |
| 28手目 | 後手 | 3 | 05時間16分21秒 | 00時間00分03秒 | |
| 29手目 | 先手 | 3 | 05時間16分23秒 | 00時間00分02秒 | |
| 30手目 | 後手 | 3 | 05時間16分23秒 | 00時間00分00秒 | |
| 31手目 | 先手 | — | —時間—分—秒 | —時間—分—秒 | |
| 32手目 | 後手 | — | —時間—分—秒 | —時間—分—秒 | |
| 33手目 | 先手 | — | —時間—分—秒 | —時間—分—秒 | |
| 34手目 | 後手 | — | —時間—分—秒 | —時間—分—秒 | |
| 35手目 | 先手 | — | —時間—分—秒 | —時間—分—秒 | |
| 36手目 | 後手 | — | —時間—分—秒 | —時間—分—秒 | |

[1] [2] [3] [4] [5] [6]

| | | | | | |
|----|----|----|----|----|----|
| | | | 25 | 23 | 20 |
| | 26 | | 17 | 9 | 18 |
| | 24 | 30 | 14 | 8 | 16 |
| 19 | 22 | 29 | 13 | 6 | 15 |
| 12 | 21 | 28 | 7 | 5 | 11 |
| 2 | 1 | 27 | 3 | 4 | 10 |

 ** 2D4M **

テヨミ

(3-9)? 9

ニンズウ

(0-2)? 0



(1-6)? 3

《第114図》 2次元4目ならべ実行用ダンプリスト

| | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| C100 | C3 | 40 | C3 | C3 | 22 | C3 | C3 | 51 | C1 | C3 | 48 | C1 | C3 | 5C | C1 | C3 |
| C110 | 64 | C1 | C3 | 6B | C1 | C3 | 75 | C1 | C3 | 74 | C1 | C3 | 90 | C1 | C3 | 04 |
| C120 | C2 | C3 | 32 | C2 | C3 | 48 | C2 | C3 | 6B | C1 | C3 | 2A | C2 | C3 | 7C | C2 |
| C130 | C3 | 9A | C2 | C3 | A0 | C2 | C3 | A5 | C2 | C3 | 71 | C1 | C3 | 26 | C3 | C3 |
| C140 | 66 | 5C | C3 | 60 | C3 | C3 | 40 | C3 | 3E | 0D | CD | 71 | C1 | 3E | 0A | 18 |
| C150 | 20 | 53 | 14 | 15 | C8 | 3E | 20 | CD | 71 | C1 | 18 | F7 | CD | 64 | C1 | D0 |
| C160 | CD | B9 | 5F | C9 | DB | 09 | FE | 7F | C0 | 37 | C9 | CD | 75 | 0F | CD | C1 |
| C170 | 5F | C3 | 2C | C3 | 43 | 78 | CD | 7E | C1 | 78 | E6 | 0F | 18 | 06 | E6 | F0 |
| C180 | 0F | 0F | 0F | 0F | FE | 0A | FA | 8B | C1 | C6 | 07 | C6 | 30 | C3 | 71 | C1 |
| C190 | C5 | CD | A3 | C1 | C1 | 7D | 81 | D6 | 87 | 57 | FA | 55 | C2 | CD | 52 | C1 |
| C1A0 | C3 | 55 | C2 | 7A | B7 | FA | AB | C1 | C3 | B5 | C1 | CD | 2A | C2 | CD | B5 |
| C1B0 | C1 | 2B | 36 | 2D | C9 | 21 | 87 | ED | 36 | 00 | E5 | 21 | 0A | 00 | CD | CD |
| C1C0 | C1 | 7D | C6 | 30 | E1 | 2B | 77 | 7B | B2 | C2 | BA | C1 | C9 | 0E | 00 | 0C |
| C1D0 | 29 | D2 | CF | C1 | CD | FD | C1 | E5 | 21 | 00 | 00 | E3 | CD | 04 | C2 | 3F |
| C1E0 | DA | E7 | C1 | EB | 19 | EB | B7 | E3 | 7D | 17 | 6F | 7C | 17 | 67 | E3 | 0D |
| C1F0 | CA | F9 | C1 | CD | FC | C1 | C3 | DC | C1 | E1 | EB | C9 | B7 | 7C | 1F | 67 |
| C200 | 7D | 1F | 6F | C9 | 7B | 95 | 5F | 7A | 9C | 57 | C9 | F2 | 14 | C2 | 04 | EB |
| C210 | CD | 2A | C2 | EB | 7A | B7 | F2 | 1D | C2 | 05 | CD | 2A | C2 | CD | CD | C1 |
| C220 | 22 | 4E | ED | 2A | 68 | ED | 3E | 00 | B8 | C8 | 7A | 2F | 57 | 7B | 2F | 5F |
| C230 | 13 | C9 | D5 | 0E | 10 | 11 | 00 | 00 | E3 | CD | FC | C1 | E3 | 30 | 03 | EB |
| C240 | 19 | EB | 29 | 0D | 20 | F2 | C1 | C9 | 06 | 00 | 7C | B7 | 20 | BD | B5 | 20 |
| C250 | C3 | 3E | 2F | 18 | 0D | AF | 47 | 7E | 23 | B8 | CA | 5C | C1 | CD | 71 | C1 |
| C260 | 18 | F5 | 3E | 07 | CD | 71 | C1 | C3 | 66 | 5C | 20 | 0A | 7B | B7 | 20 | 06 |
| C270 | 11 | 01 | 00 | F6 | FF | C9 | 11 | 00 | 00 | E6 | 00 | C9 | E5 | D5 | 2A | 52 |
| C280 | ED | 11 | 09 | 3D | CD | 32 | C2 | 13 | EB | 22 | 52 | ED | EB | E1 | 5A | 16 |
| C290 | 00 | CD | 32 | C2 | 5A | 16 | 00 | E1 | 13 | C9 | 7A | A7 | FA | 2A | C2 | C9 |
| C2A0 | 7A | A7 | C3 | 6A | C2 | 3E | 3F | CD | 57 | 02 | CD | 8A | 1B | 21 | 96 | EC |
| C2B0 | 7E | FE | 2B | 28 | 0F | FE | 2D | 28 | 12 | FE | 24 | 28 | 1B | 21 | 95 | EC |
| C2C0 | CD | FD | C2 | C9 | 21 | 96 | EC | CD | FD | C2 | C9 | 21 | 96 | EC | CD | FD |
| C2D0 | C2 | 21 | 00 | 00 | ED | 52 | EB | C9 | 11 | 97 | EC | 21 | 00 | 00 | 1A | CD |
| C2E0 | C1 | 5F | FE | 00 | 28 | 0B | CD | 39 | 5E | 38 | 15 | CD | 4B | 5E | 13 | 18 |
| C2F0 | ED | EB | C9 | E1 | 37 | 3F | 3E | 07 | CD | 57 | 02 | 18 | A8 | 11 | 00 | 00 |
| C300 | D7 | D0 | E5 | F5 | 21 | 98 | 19 | CD | 95 | 40 | DA | 1E | C3 | 62 | 6B | 19 |
| C310 | 29 | 19 | 29 | F1 | D6 | 30 | 5F | 16 | 00 | 19 | EB | E1 | 18 | E2 | F1 | E1 |
| C320 | 18 | D1 | 7B | C3 | 71 | C1 | CD | A3 | C1 | C3 | 55 | C2 | F5 | 3A | 58 | EA |
| C330 | FE | 09 | 28 | 05 | F1 | CD | 35 | 00 | C9 | F1 | CD | 2B | 00 | 18 | F6 | 00 |
| C340 | 31 | FF | FF | 21 | 00 | ED | 22 | 88 | ED | C3 | 00 | C5 | 00 | 00 | 00 | 00 |
| C350 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C360 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C370 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C380 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C390 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C3A0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C3B0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C3C0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C3D0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C3E0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C3F0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C400 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C410 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

| | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| C420 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C430 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C440 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C450 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C460 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C470 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C480 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C490 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C4A0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C4B0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C4C0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C4D0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C4E0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C4F0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| C500 | C3 | 94 | CC | 21 | 00 | 00 | 22 | 0E | ED | 21 | 00 | 00 | 22 | 14 | ED | 21 |
| C510 | 05 | 00 | EB | CD | 16 | C5 | D5 | D5 | 2A | 14 | ED | EB | 2A | 38 | ED | 19 |
| C520 | D1 | 6E | 26 | 00 | EB | 21 | 00 | 00 | EB | CD | 1E | C1 | F2 | 32 | C5 | C3 |
| C530 | A2 | C5 | 21 | 06 | 00 | 22 | 1A | ED | D5 | 2A | 14 | ED | EB | 2A | 1A | ED |
| C540 | 19 | EB | 2A | 38 | ED | 19 | D1 | 6E | 26 | 00 | EB | 21 | 00 | 00 | EB | CD |
| C550 | 1E | C1 | F2 | 58 | C5 | C3 | 73 | C5 | 2A | 1A | ED | EB | 21 | 06 | 00 | 19 |
| C560 | 22 | 1A | ED | 2A | 1A | ED | EB | 21 | 24 | 00 | CD | 1E | C1 | F2 | 73 | C5 |
| C570 | C3 | 38 | C5 | 2A | 16 | ED | EB | 21 | 02 | 00 | 19 | EB | 2A | 0E | ED | 19 |
| C580 | EB | 2A | 32 | ED | 19 | E5 | 2A | 14 | ED | EB | 2A | 1A | ED | 19 | EB | 21 |
| C590 | 06 | 00 | CD | 1E | C1 | E1 | 73 | 2A | 0E | ED | EB | 21 | 01 | 00 | 19 | 22 |
| C5A0 | 0E | ED | 2A | 14 | ED | EB | 21 | 01 | 00 | 19 | 22 | 14 | ED | D1 | D5 | CD |
| C5B0 | 1E | C1 | D1 | E1 | FA | B9 | C5 | E5 | E9 | 2A | 16 | ED | EB | 2A | 32 | ED |
| C5C0 | 19 | E5 | 2A | 0E | ED | EB | 21 | 02 | 00 | 19 | EB | E1 | 73 | C9 | 21 | 00 |
| C5D0 | 00 | 22 | 28 | ED | 21 | 00 | 00 | 22 | 1A | ED | D5 | 2A | 1A | ED | EB | 2A |
| C5E0 | 30 | ED | 19 | D1 | 6E | 26 | 00 | 22 | 1E | ED | D5 | 2A | 1A | ED | EB | 21 |
| C5F0 | 01 | 00 | 19 | EB | 2A | 30 | ED | 19 | D1 | 6E | 26 | 00 | 22 | 20 | ED | 2A |
| C600 | 1E | ED | EB | 21 | FF | 00 | CD | 1E | C1 | B3 | C2 | 1A | C6 | D5 | 21 | 01 |
| C610 | 00 | EB | CD | 2A | C1 | EB | D1 | 22 | 1E | ED | 2A | 10 | ED | 22 | 24 | ED |
| C620 | 2A | 12 | ED | 22 | 26 | ED | 21 | 01 | 00 | 22 | 0E | ED | 21 | 00 | 00 | 22 |
| C630 | 22 | ED | 2A | 24 | ED | EB | 2A | 1E | ED | 19 | 22 | 24 | ED | 2A | 26 | ED |
| C640 | EB | 2A | 20 | ED | 19 | 22 | 26 | ED | D5 | 2A | 24 | ED | EB | 21 | 05 | 00 |
| C650 | EB | CD | 1E | C1 | 11 | 00 | 00 | F2 | 5B | C6 | 1C | EB | D1 | EB | D5 | 2A |
| C660 | 24 | ED | EB | 21 | 00 | 00 | CD | 1E | C1 | 11 | 00 | 00 | F2 | 70 | C6 | 1C |
| C670 | EB | D1 | 19 | EB | D5 | 2A | 26 | ED | EB | 21 | 05 | 00 | EB | CD | 1E | C1 |
| C680 | 11 | 00 | 00 | F2 | 87 | C6 | 1C | EB | D1 | 19 | EB | 7B | B2 | CA | 93 | C6 |
| C690 | C3 | E3 | C6 | D5 | 2A | 26 | ED | EB | 21 | 06 | 00 | CD | 21 | C1 | 2A | 24 |
| C6A0 | ED | 19 | EB | 2A | 38 | ED | 19 | D1 | 6E | 26 | 00 | 22 | 14 | ED | 2A | 14 |
| C6B0 | ED | EB | 2A | 0C | ED | CD | 1E | C1 | B3 | C2 | CA | C6 | 2A | 0E | ED | EB |
| C6C0 | 21 | 01 | 00 | 19 | 22 | 0E | ED | C3 | 32 | C6 | 2A | 14 | ED | EB | 21 | 00 |
| C6D0 | 00 | CD | 1E | C1 | B3 | C2 | E3 | C6 | 2A | 22 | ED | EB | 21 | 01 | 00 | 19 |
| C6E0 | 22 | 22 | ED | 2A | 10 | ED | 22 | 24 | ED | 2A | 12 | ED | 22 | 26 | ED | 2A |
| C6F0 | 24 | ED | EB | 2A | 1E | ED | CD | 1E | C1 | EB | 22 | 24 | ED | 2A | 26 | ED |
| C700 | EB | 2A | 20 | ED | CD | 1E | C1 | EB | 22 | 26 | ED | D5 | 2A | 24 | ED | EB |
| C710 | 21 | 05 | 00 | EB | CD | 1E | C1 | 11 | 00 | 00 | F2 | 1E | C7 | 1C | EB | D1 |
| C720 | EB | D5 | 2A | 24 | ED | EB | 21 | 00 | 00 | CD | 1E | C1 | 11 | 00 | 00 | F2 |
| C730 | 33 | C7 | 1C | EB | D1 | 19 | EB | D5 | 2A | 26 | ED | EB | 21 | 00 | 00 | CD |
| C740 | 1E | C1 | 11 | 00 | 00 | F2 | 49 | C7 | 1C | EB | D1 | 19 | EB | 7B | B2 | CA |

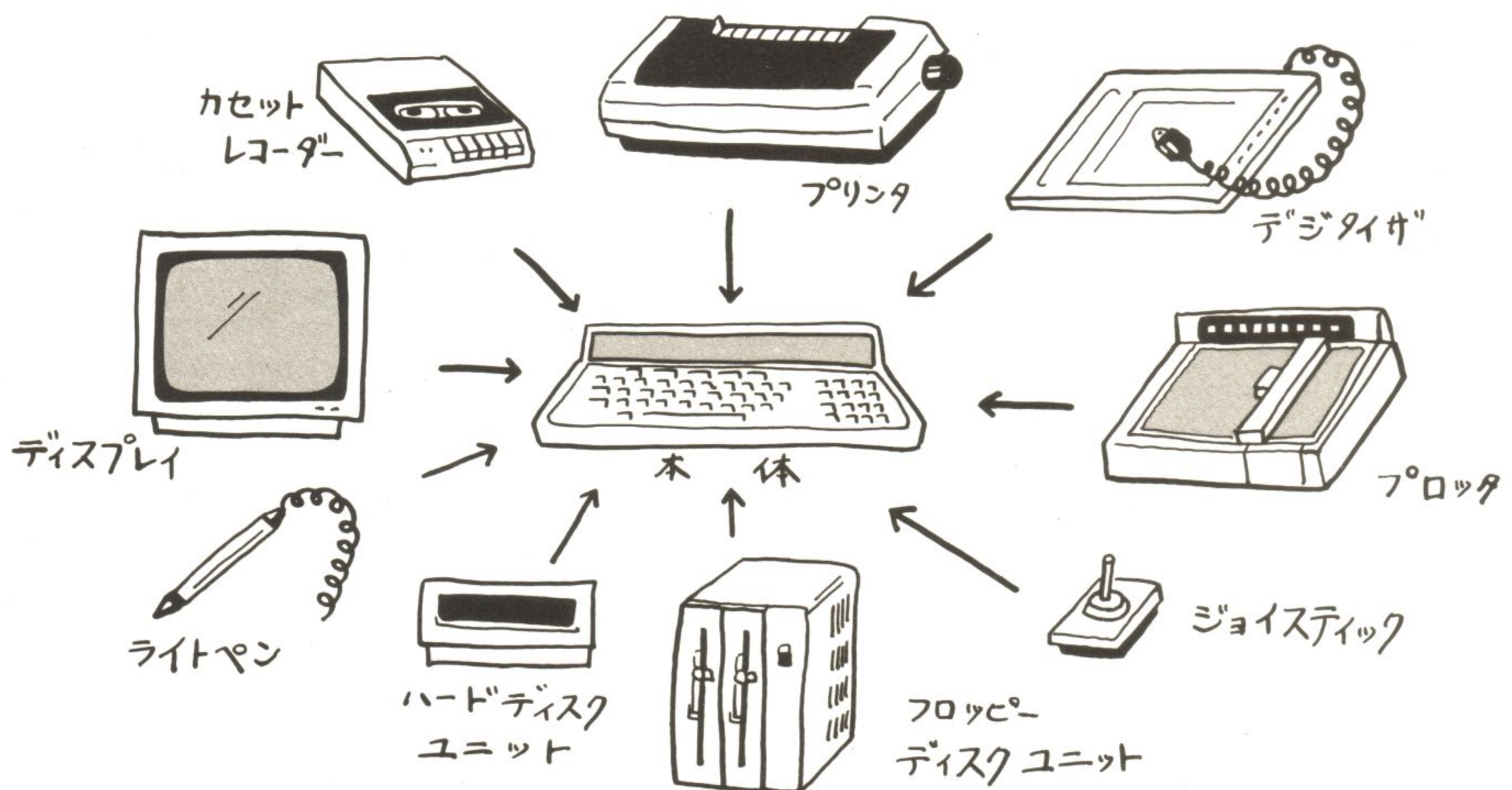
| | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| C750 | 55 | C7 | C3 | A5 | C7 | D5 | 2A | 26 | ED | EB | 21 | 06 | 00 | CD | 21 | C1 |
| C760 | 2A | 24 | ED | 19 | EB | 2A | 38 | ED | 19 | D1 | 6E | 26 | 00 | 22 | 14 | ED |
| C770 | 2A | 14 | ED | EB | 2A | 0C | ED | CD | 1E | C1 | B3 | C2 | 8C | C7 | 2A | 0E |
| C780 | ED | EB | 21 | 01 | 00 | 19 | 22 | 0E | ED | C3 | EF | C6 | 2A | 14 | ED | EB |
| C790 | 21 | 00 | 00 | CD | 1E | C1 | B3 | C2 | A5 | C7 | 2A | 22 | ED | EB | 21 | 01 |
| C7A0 | 00 | 19 | 22 | 22 | ED | 2A | 0E | ED | EB | 21 | 03 | 00 | EB | CD | 1E | C1 |
| C7B0 | F2 | BC | C7 | 21 | C8 | 00 | 22 | 28 | ED | C3 | F3 | C7 | 2A | 0E | ED | EB |
| C7C0 | 2A | 0E | ED | CD | 21 | C1 | 2A | 22 | ED | CD | 21 | C1 | 21 | 01 | 00 | 19 |
| C7D0 | EB | 2A | 28 | ED | 19 | 22 | 28 | ED | 2A | 1A | ED | EB | 21 | 02 | 00 | 19 |
| C7E0 | 22 | 1A | ED | 2A | 1A | ED | EB | 21 | 07 | 00 | CD | 1E | C1 | F2 | F3 | C7 |
| C7F0 | C3 | DA | C5 | 2A | 06 | ED | EB | 2A | 0C | ED | CD | 1E | C1 | B3 | CA | 0F |
| C800 | C8 | 21 | C9 | 00 | EB | 2A | 28 | ED | CD | 1E | C1 | EB | 22 | 28 | ED | C9 |
| C810 | 2A | 16 | ED | 22 | 18 | ED | 2A | 16 | ED | EB | 21 | 08 | 00 | CD | 1E | C1 |
| C820 | EB | 22 | 16 | ED | 21 | 03 | 00 | EB | 2A | 0C | ED | CD | 1E | C1 | EB | 22 |
| C830 | 0C | ED | 21 | 00 | 00 | 22 | 2C | ED | D5 | 2A | 16 | ED | EB | 21 | 10 | 00 |
| C840 | CD | 24 | C1 | EB | D1 | 2A | 4E | ED | EB | 21 | 08 | 00 | CD | 1E | C1 | FA |
| C850 | 55 | C8 | C3 | D0 | C8 | D5 | 2A | 18 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A |
| C860 | 32 | ED | 19 | D1 | 6E | 26 | 00 | EB | D5 | 2A | 16 | ED | EB | 21 | 01 | 00 |
| C870 | 19 | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 | 00 | EB | CD | 1E | C1 | FA | 96 |
| C880 | C8 | 2A | 18 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 | ED | 19 | E5 | 21 |
| C890 | FF | 00 | EB | E1 | 73 | C9 | 2A | 16 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A |
| C8A0 | 32 | ED | 19 | E5 | D5 | 2A | 18 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 |
| C8B0 | ED | 19 | D1 | 6E | 26 | 00 | EB | E1 | 73 | 2A | 18 | ED | EB | 21 | 01 | 00 |
| C8C0 | 19 | EB | 2A | 32 | ED | 19 | E5 | 21 | FF | 00 | EB | E1 | 73 | C3 | 47 | C9 |
| C8D0 | D5 | 2A | 18 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 | ED | 19 | D1 | 6E |
| C8E0 | 26 | 00 | EB | D5 | 2A | 16 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 | ED |
| C8F0 | 19 | D1 | 6E | 26 | 00 | CD | 1E | C1 | FA | 10 | C9 | 2A | 18 | ED | EB | 21 |
| C900 | 01 | 00 | 19 | EB | 2A | 32 | ED | 19 | E5 | 21 | 00 | 00 | EB | E1 | 73 | C9 |
| C910 | 2A | 16 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 | ED | 19 | E5 | D5 | 2A |
| C920 | 18 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 | 00 |
| C930 | EB | E1 | 73 | 2A | 18 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 | ED | 19 |
| C940 | E5 | 21 | 00 | 00 | EB | E1 | 73 | 2A | 16 | ED | EB | 21 | 00 | 00 | CD | 1E |
| C950 | C1 | B3 | C2 | 76 | C9 | D5 | 21 | 00 | 00 | EB | 2A | 32 | ED | 19 | D1 | 6E |
| C960 | 26 | 00 | 22 | 1C | ED | D5 | 2A | 1C | ED | EB | 2A | 32 | ED | 19 | D1 | 6E |
| C970 | 26 | 00 | 22 | 34 | ED | C9 | 2A | 18 | ED | EB | 21 | 10 | 00 | CD | 1E | C1 |
| C980 | EB | 22 | 18 | ED | 2A | 18 | ED | EB | 21 | 00 | 00 | CD | 1E | C1 | F2 | 92 |
| C990 | C9 | C9 | D5 | 2A | 16 | ED | EB | 21 | 10 | 00 | CD | 24 | C1 | EB | D1 | 2A |
| C9A0 | 4E | ED | EB | 21 | 08 | 00 | CD | 1E | C1 | FA | AF | C9 | C3 | E4 | C9 | D5 |
| C9B0 | 2A | 16 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 |
| C9C0 | 00 | EB | D5 | 2A | 18 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 | ED | 19 |
| C9D0 | D1 | 6E | 26 | 00 | CD | 1E | C1 | F2 | DD | C9 | C3 | 76 | C9 | 21 | 01 | 00 |
| C9E0 | 22 | 2C | ED | C9 | D5 | 2A | 16 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 |
| C9F0 | ED | 19 | D1 | 6E | 26 | 00 | EB | D5 | 2A | 18 | ED | EB | 21 | 01 | 00 | 19 |
| CA00 | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 | 00 | EB | CD | 1E | C1 | F2 | 13 | CA |
| CA10 | C3 | 76 | C9 | C3 | DD | C9 | 21 | 00 | 00 | 22 | 12 | ED | 2A | 10 | ED | EB |
| CA20 | 21 | 04 | 00 | CD | 21 | C1 | 21 | 02 | 00 | 19 | 22 | 16 | ED | D5 | 21 | 06 |
| CA30 | 00 | EB | 2A | 10 | ED | 19 | EB | 2A | 38 | ED | 19 | D1 | 6E | 26 | 00 | EB |
| CA40 | 21 | 00 | 00 | EB | CD | 1E | C1 | F2 | 4D | CA | C3 | 17 | CB | 11 | 00 | 00 |
| CA50 | CD | 53 | CA | D5 | 2A | 12 | ED | EB | 21 | 03 | 00 | CD | 21 | C1 | 21 | 06 |
| CA60 | 00 | 19 | 22 | 18 | ED | 2A | 12 | ED | EB | 21 | 03 | 00 | CD | 21 | C1 | 21 |
| CA70 | 08 | 00 | 19 | EB | CD | 77 | CA | D5 | CD | 46 | CB | 21 | 01 | 00 | 22 | 1C |

| | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CA80 | ED | 21 | F4 | 01 | EB | CD | 88 | CA | D5 | 2A | 1C | ED | EB | 21 | 01 | 00 |
| CA90 | 19 | 22 | 1C | ED | D1 | D5 | CD | 1E | C1 | D1 | E1 | FA | A0 | CA | E5 | E9 |
| CAA0 | CD | 36 | CC | 2A | 18 | ED | EB | 21 | 01 | 00 | 19 | 22 | 18 | ED | D1 | D5 |
| CAB0 | CD | 1E | C1 | D1 | E1 | FA | BA | CA | E5 | E9 | 2A | 12 | ED | EB | 21 | 01 |
| CAC0 | 00 | 19 | 22 | 12 | ED | D5 | D5 | 2A | 12 | ED | EB | 21 | 01 | 00 | 19 | EB |
| CAD0 | 21 | 06 | 00 | CD | 21 | C1 | 2A | 10 | ED | 19 | EB | 2A | 38 | ED | 19 | D1 |
| CAE0 | 6E | 26 | 00 | EB | 21 | 00 | 00 | EB | CD | 1E | C1 | 11 | 00 | 00 | F2 | F2 |
| CAF0 | CA | 1C | EB | D1 | EB | D5 | 2A | 12 | ED | EB | 21 | 04 | 00 | EB | CD | 1E |
| CB00 | C1 | 11 | 00 | 00 | F2 | 08 | CB | 1C | EB | D1 | 19 | D1 | D5 | CD | 1E | C1 |
| CB10 | D1 | E1 | FA | 17 | CB | E5 | E9 | 2A | 12 | ED | EB | 21 | 03 | 00 | CD | 21 |
| CB20 | C1 | 21 | 06 | 00 | 19 | 22 | 18 | ED | CD | 46 | CB | 2A | 12 | ED | EB | 21 |
| CB30 | 06 | 00 | CD | 21 | C1 | 2A | 10 | ED | 19 | EB | 2A | 38 | ED | 19 | E5 | 2A |
| CB40 | 06 | ED | EB | E1 | 73 | C9 | 2A | 06 | ED | EB | 21 | 01 | 00 | CD | 1E | C1 |
| CB50 | B3 | C2 | 63 | CB | 21 | 00 | 00 | EB | 2A | 2E | ED | 19 | E5 | 21 | C8 | 00 |
| CB60 | EB | E1 | 73 | 2A | 06 | ED | EB | 21 | 02 | 00 | CD | 1E | C1 | B3 | C2 | 80 |
| CB70 | CB | 21 | 00 | 00 | EB | 2A | 2E | ED | 19 | E5 | 21 | A8 | 00 | EB | E1 | 73 |
| CB80 | 21 | 01 | 00 | EB | 2A | 36 | ED | 19 | E5 | 2A | 16 | ED | EB | E1 | 73 | 21 |
| CB90 | 00 | 00 | EB | 2A | 36 | ED | 19 | E5 | 2A | 18 | ED | EB | E1 | 73 | C3 | A4 |
| CBA0 | CB | 86 | 86 | 86 | 06 | 03 | 21 | A1 | CB | 7E | CD | 39 | C1 | 23 | 10 | F9 |
| CBB0 | 21 | 01 | 00 | EB | 2A | 36 | ED | 19 | E5 | 2A | 16 | ED | EB | E1 | 73 | 21 |
| CBC0 | 00 | 00 | EB | 2A | 36 | ED | 19 | E5 | 2A | 18 | ED | EB | 21 | 01 | 00 | 19 |
| CBD0 | EB | E1 | 73 | C3 | D9 | CB | 87 | 87 | 87 | 06 | 03 | 21 | D6 | CB | 7E | CD |
| CBE0 | 39 | C1 | 23 | 10 | F9 | 21 | 01 | 00 | EB | 2A | 36 | ED | 19 | E5 | 2A | 16 |
| CBF0 | ED | EB | E1 | 73 | 21 | 00 | 00 | EB | 2A | 36 | ED | 19 | E5 | 2A | 18 | ED |
| CC00 | EB | 21 | 02 | 00 | 19 | EB | E1 | 73 | C3 | 0E | CC | 87 | 87 | 87 | 06 | 03 |
| CC10 | 21 | 0B | CC | 7E | CD | 39 | C1 | 23 | 10 | F9 | 21 | 00 | 00 | EB | 2A | 2E |
| CC20 | ED | 19 | E5 | 21 | E8 | 00 | EB | E1 | 73 | C9 | 21 | 24 | 00 | 22 | 16 | ED |
| CC30 | 21 | 14 | 00 | 22 | 18 | ED | 21 | 01 | 00 | EB | 2A | 36 | ED | 19 | E5 | 2A |
| CC40 | 16 | ED | EB | E1 | 73 | 21 | 00 | 00 | EB | 2A | 36 | ED | 19 | E5 | 2A | 18 |
| CC50 | ED | EB | E1 | 73 | C3 | 5A | CC | 20 | 20 | 20 | 06 | 03 | 21 | 57 | CC | 7E |
| CC60 | CD | 39 | C1 | 23 | 10 | F9 | C9 | 3E | 20 | D3 | 40 | 21 | 01 | 00 | 22 | 1C |
| CC70 | ED | 21 | F4 | 01 | EB | CD | 78 | CC | D5 | 2A | 1C | ED | EB | 21 | 01 | 00 |
| CC80 | 19 | 22 | 1C | ED | D1 | D5 | CD | 1E | C1 | D1 | E1 | FA | 90 | CC | E5 | E9 |
| CC90 | AF | D3 | 40 | C9 | 01 | 19 | 28 | CD | 3A | 09 | 21 | 18 | 01 | 22 | 5D | EA |
| CCA0 | 01 | FF | 00 | CD | F7 | 08 | 21 | 00 | C4 | 22 | 38 | ED | 2A | 38 | ED | EB |
| CCB0 | 21 | 24 | 00 | 19 | 22 | 30 | ED | 2A | 30 | ED | EB | 21 | 08 | 00 | 19 | 22 |
| CCC0 | 32 | ED | 21 | 63 | EA | 22 | 36 | ED | 21 | 5B | EA | 22 | 2E | ED | 21 | 01 |
| CCD0 | 00 | EB | 2A | 2E | ED | 19 | E5 | 21 | 50 | 00 | EB | E1 | 73 | 21 | 00 | 00 |
| CCE0 | EB | 2A | 30 | ED | 19 | E5 | 21 | 01 | 00 | EB | E1 | 73 | 21 | 01 | 00 | EB |
| CCF0 | 2A | 30 | ED | 19 | E5 | 21 | 00 | 00 | EB | E1 | 73 | 21 | 02 | 00 | EB | 2A |
| CD00 | 30 | ED | 19 | E5 | 21 | 01 | 00 | EB | E1 | 73 | 21 | 03 | 00 | EB | 2A | 30 |
| CD10 | ED | 19 | E5 | 21 | 01 | 00 | EB | E1 | 73 | 21 | 04 | 00 | EB | 2A | 30 | ED |
| CD20 | 19 | E5 | 21 | 00 | 00 | EB | E1 | 73 | 21 | 05 | 00 | EB | 2A | 30 | ED | 19 |
| CD30 | E5 | 21 | 01 | 00 | EB | E1 | 73 | 21 | 06 | 00 | EB | 2A | 30 | ED | 19 | E5 |
| CD40 | D5 | 21 | 01 | 00 | EB | CD | 2A | C1 | EB | D1 | EB | E1 | 73 | 21 | 07 | 00 |
| CD50 | EB | 2A | 30 | ED | 19 | E5 | 21 | 01 | 00 | EB | E1 | 73 | 21 | 00 | 00 | 22 |
| CD60 | 1C | ED | 21 | 23 | 00 | EB | CD | 69 | CD | D5 | 2A | 1C | ED | EB | 2A | 38 |
| CD70 | ED | 19 | E5 | 21 | 00 | 00 | EB | E1 | 73 | 2A | 1C | ED | EB | 21 | 01 | 00 |
| CD80 | 19 | 22 | 1C | ED | D1 | D5 | CD | 1E | C1 | D1 | E1 | FA | 90 | CD | E5 | E9 |
| CD90 | 21 | 0C | 00 | EB | 7B | CD | 03 | C1 | 21 | 1B | 00 | EB | CD | 06 | C1 | C3 |
| CDA0 | AC | CD | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 2A | 06 | 0A | 21 | A2 |

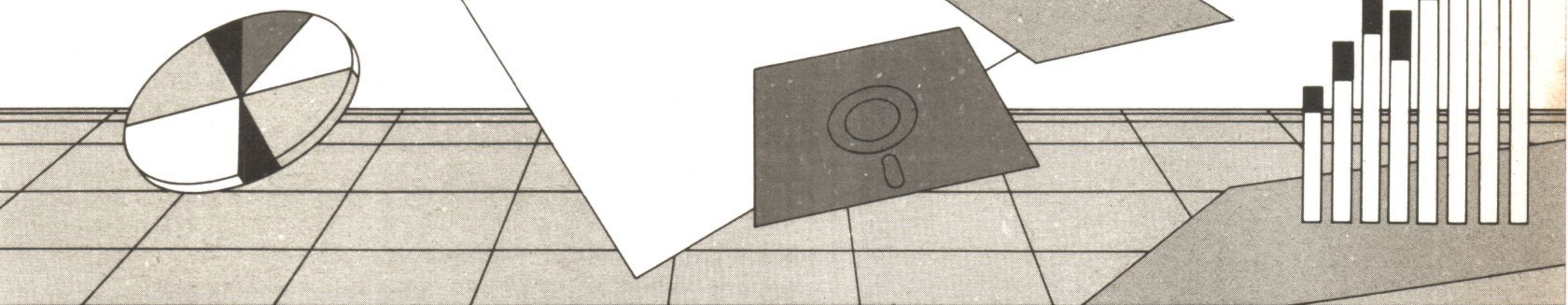
| | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CDB0 | CD | 7E | CD | 39 | C1 | 23 | 10 | F9 | CD | 09 | C1 | 21 | 1B | 00 | EB | CD |
| CDC0 | 06 | C1 | C3 | CF | CD | 2A | 2A | 20 | 32 | 44 | 34 | 4D | 20 | 2A | 2A | 06 |
| CDD0 | 0A | 21 | C5 | CD | 7E | CD | 39 | C1 | 23 | 10 | F9 | CD | 09 | C1 | C3 | 06 |
| CDE0 | CE | 20 | 5B | 31 | 5D | 20 | 5B | 32 | 5D | 20 | 5B | 33 | 5D | 20 | 5B | 34 |
| CDF0 | 5D | 20 | 5B | 35 | 5D | 20 | 5B | 36 | 5D | 20 | 20 | 20 | 2A | 2A | 2A | 2A |
| CE00 | 2A | 2A | 2A | 2A | 2A | 2A | 06 | 25 | 21 | E1 | CD | 7E | CD | 39 | C1 | 23 |
| CE10 | 10 | F9 | CD | 09 | C1 | CD | 09 | C1 | C3 | 34 | CE | 87 | 87 | 87 | 87 | 87 |
| CE20 | 87 | 87 | 87 | 87 | 87 | 87 | 87 | 87 | 87 | 87 | 87 | 87 | 87 | 87 | 87 | 87 |
| CE30 | 87 | 87 | 87 | 87 | 06 | 19 | 21 | 1B | CE | 7E | CD | 39 | C1 | 23 | 10 | F9 |
| CE40 | CD | 09 | C1 | 21 | 01 | 00 | 22 | 1C | ED | 21 | 12 | 00 | EB | CD | 50 | CE |
| CE50 | D5 | C3 | 6D | CE | 87 | 20 | 20 | 20 | 87 | 20 | 20 | 20 | 87 | 20 | 20 | 20 |
| CE60 | 87 | 20 | 20 | 20 | 87 | 20 | 20 | 20 | 87 | 20 | 20 | 20 | 87 | 06 | 19 | 21 |
| CE70 | 54 | CE | 7E | CD | 39 | C1 | 23 | 10 | F9 | CD | 09 | C1 | 2A | 1C | ED | EB |
| CE80 | 21 | 01 | 00 | 19 | 22 | 1C | ED | D1 | D5 | CD | 1E | C1 | D1 | E1 | FA | 93 |
| CE90 | CE | E5 | E9 | C3 | AF | CE | 87 | 86 | 86 | 86 | 87 | 86 | 86 | 86 | 87 | 86 |
| CEA0 | 86 | 86 | 87 | 86 | 86 | 86 | 87 | 86 | 86 | 86 | 87 | 86 | 86 | 86 | 87 | 06 |
| CEB0 | 19 | 21 | 96 | CE | 7E | CD | 39 | C1 | 23 | 10 | F9 | 21 | 01 | 00 | EB | 2A |
| CEC0 | 36 | ED | 19 | E5 | 21 | 1C | 00 | EB | E1 | 73 | 21 | 00 | 00 | EB | 2A | 36 |
| CED0 | ED | 19 | E5 | 21 | 06 | 00 | EB | E1 | 73 | C3 | DF | CE | C3 | D6 | D0 | 06 |
| CEE0 | 03 | 21 | DC | CE | 7E | CD | 39 | C1 | 23 | 10 | F9 | 21 | 01 | 00 | EB | 2A |
| CEF0 | 36 | ED | 19 | E5 | 21 | 1E | 00 | EB | E1 | 73 | 21 | 00 | 00 | EB | 2A | 36 |
| CF00 | ED | 19 | E5 | 21 | 08 | 00 | EB | E1 | 73 | C3 | 13 | CF | 28 | 33 | 2D | 39 |
| CF10 | 29 | 3F | 20 | 06 | 07 | 21 | 0C | CF | 7E | CD | 39 | C1 | 23 | 10 | F9 | CD |
| CF20 | 67 | CC | CD | 12 | C1 | 6F | 26 | 00 | 22 | 2A | ED | 2A | 2A | ED | EB | 21 |
| CF30 | 03 | 00 | CD | 1E | C1 | B3 | C2 | 3C | CF | C3 | 66 | 5C | 2A | 2A | ED | EB |
| CF40 | 21 | 30 | 00 | CD | 1E | C1 | EB | 22 | 2A | ED | D5 | 2A | 2A | ED | EB | 21 |
| CF50 | 03 | 00 | CD | 1E | C1 | 11 | 00 | 00 | F2 | 5C | CF | 1C | EB | D1 | EB | D5 |
| CF60 | 2A | 2A | ED | EB | 21 | 09 | 00 | EB | CD | 1E | C1 | 11 | 00 | 00 | F2 | 72 |
| CF70 | CF | 1C | EB | D1 | 19 | EB | 7B | B2 | CA | 7E | CF | C3 | EB | CE | 2A | 2A |
| CF80 | ED | EB | 21 | 08 | 00 | CD | 21 | C1 | 21 | 10 | 00 | CD | 1E | C1 | EB | 22 |
| CF90 | 2A | ED | 21 | 01 | 00 | EB | 2A | 36 | ED | 19 | E5 | 21 | 1C | 00 | EB | E1 |
| CFA0 | 73 | 21 | 00 | 00 | EB | 2A | 36 | ED | 19 | E5 | 21 | 0A | 00 | EB | E1 | 73 |
| CFB0 | C3 | B8 | CF | C6 | DD | BD | DE | B3 | 06 | 05 | 21 | B3 | CF | 7E | CD | 39 |
| CFC0 | C1 | 23 | 10 | F9 | 21 | 01 | 00 | EB | 2A | 36 | ED | 19 | E5 | 21 | 1E | 00 |
| CFD0 | EB | E1 | 73 | 21 | 00 | 00 | EB | 2A | 36 | ED | 19 | E5 | 21 | 0C | 00 | EB |
| CFE0 | E1 | 73 | C3 | EC | CF | 28 | 30 | 2D | 32 | 29 | 3F | 20 | 06 | 07 | 21 | E5 |
| CFF0 | CF | 7E | CD | 39 | C1 | 23 | 10 | F9 | CD | 67 | CC | CD | 12 | C1 | 6F | 26 |
| D000 | 00 | 22 | 0A | ED | 2A | 0A | ED | EB | 21 | 03 | 00 | CD | 1E | C1 | B3 | C2 |
| D010 | 15 | D0 | C3 | 66 | 5C | 2A | 0A | ED | EB | 21 | 30 | 00 | CD | 1E | C1 | EB |
| D020 | 22 | 0A | ED | D5 | 2A | 0A | ED | EB | 21 | 00 | 00 | CD | 1E | C1 | 11 | 00 |
| D030 | 00 | F2 | 35 | D0 | 1C | EB | D1 | EB | D5 | 2A | 0A | ED | EB | 21 | 02 | 00 |
| D040 | EB | CD | 1E | C1 | 11 | 00 | 00 | F2 | 4B | D0 | 1C | EB | D1 | 19 | EB | 7B |
| D050 | B2 | CA | 57 | D0 | C3 | C4 | CF | 21 | 01 | 00 | 22 | 06 | ED | 2A | 0A | ED |
| D060 | EB | 21 | 01 | 00 | CD | 1E | C1 | B3 | CA | 6E | D0 | C3 | 20 | D1 | 21 | 01 |
| D070 | 00 | EB | 2A | 36 | ED | 19 | E5 | 21 | 1C | 00 | EB | E1 | 73 | 21 | 00 | 00 |
| D080 | EB | 2A | 36 | ED | 19 | E5 | 21 | 0E | 00 | EB | E1 | 73 | C3 | 92 | D0 | BE |
| D090 | DD | C3 | 06 | 03 | 21 | 8F | D0 | 7E | CD | 39 | C1 | 23 | 10 | F9 | 21 | 01 |
| DOA0 | 00 | EB | 2A | 36 | ED | 19 | E5 | 21 | 1E | 00 | EB | E1 | 73 | 21 | 00 | 00 |
| DOB0 | EB | 2A | 36 | ED | 19 | E5 | 21 | 10 | 00 | EB | E1 | 73 | C3 | C6 | D0 | 28 |
| DOC0 | 59 | 2F | 4E | 29 | 3F | 20 | 06 | 07 | 21 | BF | D0 | 7E | CD | 39 | C1 | 23 |
| DOD0 | 10 | F9 | CD | 67 | CC | CD | 12 | C1 | 6F | 26 | 00 | 22 | 1C | ED | 2A | 1C |

| | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| D0E0 | ED | EB | 21 | 03 | 00 | CD | 1E | C1 | B3 | C2 | EF | D0 | C3 | 66 | 5C | 2A |
| D0F0 | 1C | ED | EB | 21 | 59 | 00 | CD | 1E | C1 | B3 | C2 | 06 | D1 | 21 | 02 | 00 |
| D100 | 22 | 06 | ED | C3 | 20 | D1 | 2A | 1C | ED | EB | 21 | 4E | 00 | CD | 1E | C1 |
| D110 | B3 | C2 | 1D | D1 | 21 | 01 | 00 | 22 | 06 | ED | C3 | 20 | D1 | C3 | 9E | D0 |
| D120 | 21 | 00 | 00 | 22 | 08 | ED | 21 | 03 | 00 | EB | 2A | 06 | ED | CD | 1E | C1 |
| D130 | EB | 22 | 06 | ED | 2A | 08 | ED | EB | 21 | 01 | 00 | 19 | 22 | 08 | ED | D5 |
| D140 | 2A | 0A | ED | EB | 21 | 01 | 00 | CD | 1E | C1 | B3 | 11 | 00 | 00 | C2 | 52 |
| D150 | D1 | 1C | EB | D1 | EB | D5 | 2A | 06 | ED | EB | 21 | 02 | 00 | CD | 1E | C1 |
| D160 | B3 | 11 | 00 | 00 | C2 | 68 | D1 | 1C | EB | D1 | CD | 21 | C1 | D5 | 2A | 0A |
| D170 | ED | EB | 21 | 00 | 00 | CD | 1E | C1 | B3 | 11 | 00 | 00 | C2 | 80 | D1 | 1C |
| D180 | EB | D1 | 19 | EB | 7B | B2 | CA | 8C | D1 | C3 | 0C | D3 | 21 | 1C | 00 | 22 |
| D190 | 16 | ED | 21 | 13 | 00 | 22 | 18 | ED | CD | 46 | CB | CD | 2A | CC | 21 | 01 |
| D1A0 | 00 | EB | 2A | 36 | ED | 19 | E5 | 21 | 1E | 00 | EB | E1 | 73 | 21 | 00 | 00 |
| D1B0 | EB | 2A | 36 | ED | 19 | E5 | 21 | 17 | 00 | EB | E1 | 73 | C3 | C6 | D1 | 28 |
| D1C0 | 31 | 2D | 36 | 29 | 3F | 20 | 06 | 07 | 21 | BF | D1 | 7E | CD | 39 | C1 | 23 |
| D1D0 | 10 | F9 | CD | 67 | CC | CD | 12 | C1 | 6F | 26 | 00 | 22 | 34 | ED | 2A | 34 |
| D1E0 | ED | EB | 21 | 03 | 00 | CD | 1E | C1 | B3 | C2 | EF | D1 | C3 | 66 | 5C | 2A |
| D1F0 | 34 | ED | EB | 21 | 31 | 00 | CD | 1E | C1 | EB | 22 | 34 | ED | D5 | 2A | 34 |
| D200 | ED | EB | 21 | 00 | 00 | CD | 1E | C1 | 11 | 00 | 00 | F2 | 0F | D2 | 1C | EB |
| D210 | D1 | EB | D5 | 2A | 34 | ED | EB | 21 | 05 | 00 | EB | CD | 1E | C1 | 11 | 00 |
| D220 | 00 | F2 | 25 | D2 | 1C | EB | D1 | 19 | EB | D5 | D5 | 2A | 34 | ED | EB | 2A |
| D230 | 38 | ED | 19 | D1 | 6E | 26 | 00 | EB | 21 | 00 | 00 | EB | CD | 1E | C1 | 11 |
| D240 | 00 | 00 | F2 | 46 | D2 | 1C | EB | D1 | 19 | EB | 7B | B2 | CA | 52 | D2 | C3 |
| D250 | 9B | D1 | 2A | 34 | ED | 22 | 10 | ED | CD | 16 | CA | 2A | 06 | ED | 22 | 0C |
| D260 | ED | CD | CE | C5 | D5 | 2A | 28 | ED | EB | 21 | C8 | 00 | CD | 1E | C1 | 11 |
| D270 | 00 | 00 | F2 | 76 | D2 | 1C | EB | D1 | EB | D5 | 2A | 08 | ED | EB | 21 | 24 |
| D280 | 00 | CD | 1E | C1 | 11 | 00 | 00 | F2 | 8B | D2 | 1C | EB | D1 | CD | 21 | C1 |
| D290 | 7B | B2 | CA | 98 | D2 | C3 | 26 | D1 | 21 | 1C | 00 | 22 | 16 | ED | 21 | 13 |
| D2A0 | 00 | 22 | 18 | ED | CD | 46 | CB | CD | 2A | CC | 21 | 01 | 00 | EB | 2A | 36 |
| D2B0 | ED | 19 | E5 | 21 | 1E | 00 | EB | E1 | 73 | 21 | 00 | 00 | EB | 2A | 36 | ED |
| D2C0 | 19 | E5 | 21 | 17 | 00 | EB | E1 | 73 | C3 | D2 | D2 | C9 | 20 | B6 | C1 | 21 |
| D2D0 | 21 | 20 | 06 | 07 | 21 | CB | D2 | 7E | CD | 39 | C1 | 23 | 10 | F9 | CD | 12 |
| D2E0 | C1 | 6F | 26 | 00 | 22 | 1C | ED | 2A | 1C | ED | EB | 21 | 03 | 00 | CD | 1E |
| D2F0 | C1 | B3 | C2 | F8 | D2 | C3 | 66 | 5C | 2A | 1C | ED | EB | 21 | 0D | 00 | CD |
| D300 | 1E | C1 | B3 | CA | 09 | D3 | C3 | A7 | D2 | C3 | 5C | CD | 2A | 08 | ED | EB |
| D310 | 21 | 04 | 00 | CD | 1E | C1 | F2 | 33 | D3 | D5 | C5 | 21 | 06 | 00 | EB | CD |
| D320 | 2D | C1 | EB | C1 | D1 | EB | 21 | 01 | 00 | CD | 1E | C1 | EB | 22 | 34 | ED |
| D330 | C3 | 52 | D2 | 21 | 00 | 00 | 22 | 16 | ED | 2A | 06 | ED | 22 | 0C | ED | 21 |
| D340 | 01 | 00 | 22 | 1C | ED | 2A | 2A | ED | EB | 21 | 09 | 00 | 19 | EB | CD | 51 |
| D350 | D3 | D5 | 2A | 1C | ED | EB | 2A | 32 | ED | 19 | E5 | 21 | 00 | 00 | EB | E1 |
| D360 | 73 | 2A | 1C | ED | EB | 21 | 08 | 00 | 19 | EB | 2A | 32 | ED | 19 | E5 | 21 |
| D370 | FF | 00 | EB | E1 | 73 | 2A | 1C | ED | EB | 21 | 10 | 00 | 19 | 22 | 1C | ED |
| D380 | D1 | D5 | CD | 1E | C1 | D1 | E1 | FA | 8C | D3 | E5 | E9 | 11 | 00 | 00 | CD |
| D390 | 92 | D3 | D5 | D5 | 2A | 16 | ED | EB | 2A | 2A | ED | EB | CD | 1E | C1 | 11 |
| D3A0 | 00 | 00 | F2 | A6 | D3 | 1C | EB | D1 | EB | D5 | 2A | 28 | ED | EB | 21 | 01 |
| D3B0 | 00 | CD | 1E | C1 | B3 | 11 | 00 | 00 | C2 | BC | D3 | 1C | EB | D1 | 19 | EB |
| D3C0 | D5 | 2A | 28 | ED | EB | 21 | C8 | 00 | CD | 1E | C1 | B3 | 11 | 00 | 00 | C2 |
| D3D0 | D3 | D3 | 1C | EB | D1 | 19 | EB | 7B | B2 | CA | DF | D3 | C3 | C9 | D4 | CD |
| D3E0 | 03 | C5 | D5 | 2A | 16 | ED | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 | 00 | EB |
| D3F0 | 21 | 03 | 00 | CD | 1E | C1 | F2 | FC | D3 | C3 | C9 | D4 | 2A | 16 | ED | EB |
| D400 | 2A | 32 | ED | 19 | E5 | D5 | 2A | 16 | ED | EB | 2A | 32 | ED | 19 | D1 | 6E |

| | | | | | | | | | | | | | | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| D410 | 26 | 00 | EB | 21 | 01 | 00 | CD | 1E | C1 | E1 | 73 | D5 | 2A | 16 | ED | EB |
| D420 | 2A | 32 | ED | 19 | D1 | 6E | 26 | 00 | 22 | 1C | ED | D5 | 2A | 16 | ED | EB |
| D430 | 2A | 1C | ED | 19 | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 | 00 | 22 | 1C | ED |
| D440 | 2A | 1C | ED | EB | 2A | 38 | ED | 19 | E5 | 2A | 0C | ED | EB | E1 | 73 | D5 |
| D450 | 2A | 16 | ED | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 | 00 | 22 | 1C | ED | D5 |
| D460 | 2A | 16 | ED | EB | 2A | 1C | ED | 19 | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 |
| D470 | 00 | EB | 21 | 06 | 00 | CD | 24 | C1 | EB | 22 | 12 | ED | D5 | D5 | 2A | 16 |
| D480 | ED | EB | 2A | 1C | ED | 19 | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 | 00 | EB |
| D490 | 21 | 06 | 00 | CD | 24 | C1 | EB | D1 | 2A | 4E | ED | 22 | 10 | ED | CD | CE |
| D4A0 | C5 | 2A | 16 | ED | EB | 21 | 08 | 00 | 19 | 22 | 16 | ED | 21 | 03 | 00 | EB |
| D4B0 | 2A | 0C | ED | CD | 1E | C1 | EB | 22 | 0C | ED | 21 | 00 | 00 | D1 | D5 | CD |
| D4C0 | 1E | C1 | D1 | E1 | FA | C9 | D4 | E5 | E9 | D5 | 2A | 16 | ED | EB | 21 | 01 |
| D4D0 | 00 | 19 | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 | 00 | 22 | 1C | ED | D5 | 2A |
| D4E0 | 1C | ED | EB | 21 | 00 | 00 | CD | 1E | C1 | B3 | 11 | 00 | 00 | C2 | F1 | D4 |
| D4F0 | 1C | EB | D1 | EB | D5 | 2A | 1C | ED | EB | 21 | FF | 00 | CD | 1E | C1 | B3 |
| D500 | 11 | 00 | 00 | C2 | 07 | D5 | 1C | EB | D1 | 19 | EB | 7B | B2 | CA | 24 | D5 |
| D510 | 2A | 16 | ED | EB | 21 | 01 | 00 | 19 | EB | 2A | 32 | ED | 19 | E5 | 2A | 28 |
| D520 | ED | EB | E1 | 73 | 21 | 00 | 00 | 22 | 28 | ED | 2A | 16 | ED | EB | 21 | 00 |
| D530 | 00 | CD | 1E | C1 | B3 | C2 | 5D | D5 | 21 | 01 | 00 | D1 | D5 | CD | 1E | C1 |
| D540 | D1 | E1 | FA | 47 | D5 | E5 | E9 | D5 | 2A | 34 | ED | EB | 21 | 06 | 00 | CD |
| D550 | 24 | C1 | EB | D1 | 2A | 4E | ED | 22 | 34 | ED | C3 | 52 | D2 | CD | 10 | C8 |
| D560 | D5 | 2A | 16 | ED | EB | 2A | 32 | ED | 19 | D1 | 6E | 26 | 00 | 22 | 1C | ED |
| D570 | D5 | 2A | 1C | ED | EB | 2A | 16 | ED | 19 | EB | 2A | 32 | ED | 19 | D1 | 6E |
| D580 | 26 | 00 | 22 | 1C | ED | 2A | 1C | ED | EB | 2A | 38 | ED | 19 | E5 | 21 | 00 |
| D590 | 00 | EB | E1 | 73 | 2A | 2C | ED | EB | 21 | 01 | 00 | CD | 1E | C1 | B3 | C2 |
| D5A0 | B1 | D5 | 2A | 16 | ED | EB | 2A | 32 | ED | 19 | E5 | 21 | 02 | 00 | EB | E1 |
| D5B0 | 73 | C3 | E2 | D3 | | | | | | | | | | | | |



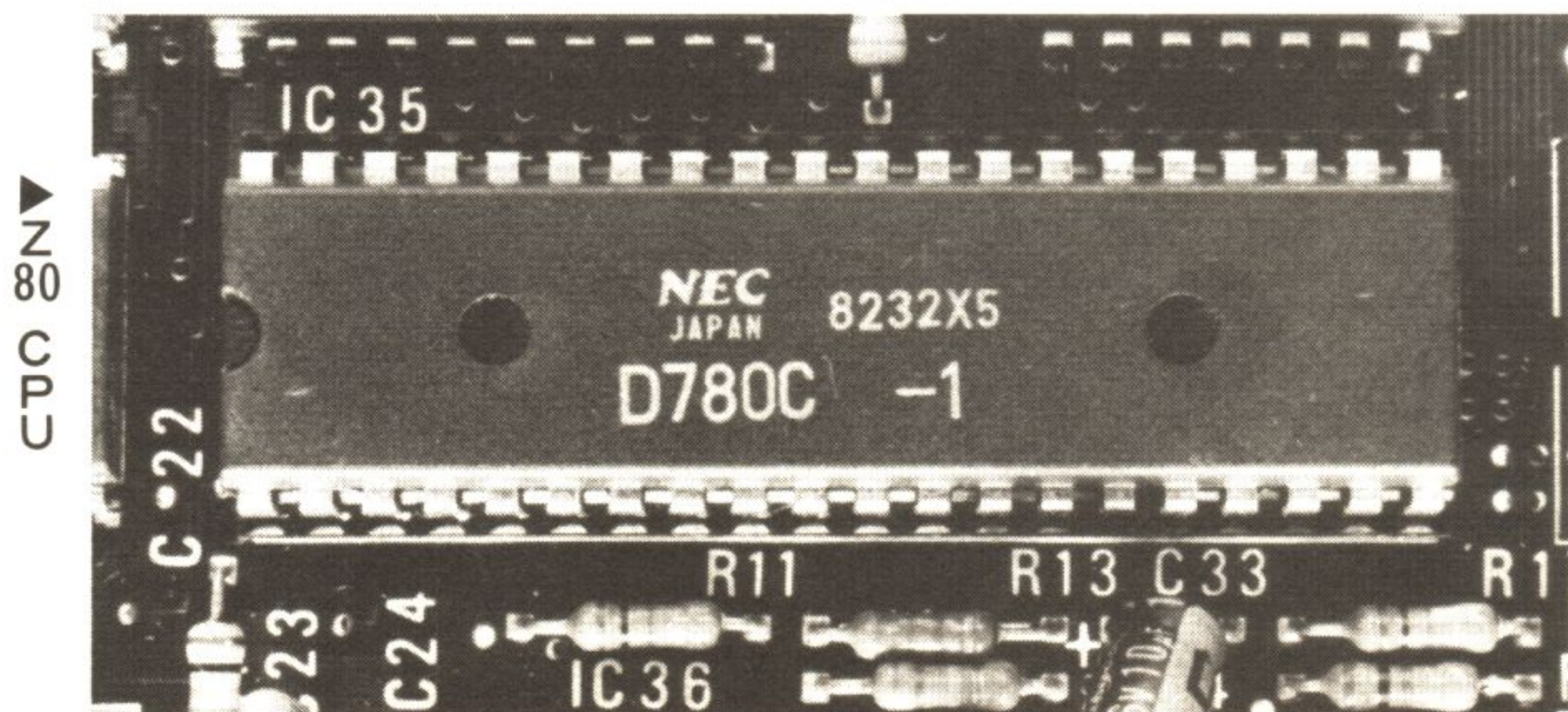
汎用サブルーチン・パッケージ アセンブル・リスト



《第115図》 汎用サブルーチン・パッケージの基本仕様

| サブルーチン | 仕 様 |
|----------|---|
| 「RUNO」 | Z80のスタック・ポインタ（SP）にFFFFHを設定後，C500H番地以降のターゲット・プログラムを実行する。 |
| 「CHARO」 | Eレジスタに格納されたキャラクタ・コードに従ったキャラクタを出力機器へ出力する。 ただし，EA58H番地の最上位ビット（MSB）が0のときCRT画面を，1のときCRT画面とプリンタを，それぞれ出力機器として選択する。 |
| 「TABO」 | Eレジスタに与えられた数のスペース（空白）コードを出力機器へ出力する。 ただし，EA58H番地の最上位ビット（MSB）が0のときCRT画面を，1のときCRT画面とプリンタを，それぞれ出力機器として選択する。 |
| 「CRLF0」 | キャリッジ・リターン・コード（0DH）およびライン・フィード・コード（0AH）を周辺機器へ出力する。 ただし，EA58H番地の最上位ビット（MSB）が0のときCRT画面を，1のときCRT画面とプリンタを，それぞれ出力機器として選択する。 |
| 「ESCO」 | ESCキーが押下されている場合には次のキー入力を待ち，入力データをAレジスタに格納してもどる。 |
| 「INKEYO」 | キーボードからAレジスタへの1文字入力を待ち，英小文字であれば英大文字に変換し，出力機器へエコーバックする。 ただし，EA58H番地の最上位ビット（MSB）が0のときCRT画面を，1のときCRT画面とプリンタを，それぞれ出力機器として選択する。 |
| 「HFOURO」 | Bレジスタに格納されたデータを2桁の16進数として周辺機器へ出力する。 ただし，EA58H番地の最上位ビット（MSB）が0のときCRT画面を，1のときCRT画面とプリンタを，それぞれ出力機器として選択する。 |
| 「HTWOO」 | Eレジスタに格納されたデータを2桁の16進数として周辺機器へ出力する。 ただし，EA58H番地の最上位ビット（MSB）が0のときCRT画面を，1のときCRT画面とプリンタを，それぞれ出力機器として選択する。 |

| | |
|----------|--|
| 「USINGO」 | DEレジスタ対に格納された有符号の16ビット整数データをCレジスタで指定する桁数の10進数として周辺機器へ右づめ出力する。 ただし、EA58H番地の最上位ビット(MSB)が0のときCRT画面を、1のときCRT画面とプリンタを、それぞれ出力機器として選択する。 |
| 「ISUBO」 | DEレジスタ対に格納された有符号16ビット長の整数を被減数とし、HLレジスタ対に格納された有符号16ビット長の整数を減数として両者の間で減算を行い、演算結果をDEレジスタ対に格納する。 |
| 「IMULTO」 | DEレジスタ対に格納された有符号16ビット長の整数を被乗数とし、HLレジスタ対に格納された有符号16ビット長の整数を乗数として両者の間で乗算を行い、演算結果をDEレジスタ対に格納する。 |
| 「IDIVO」 | DEレジスタ対に格納された有符号16ビット長の整数を被除数とし、HLレジスタ対に格納された有符号16ビット長の整数を除数として両者の間で除算を行い、演算結果をDEレジスタ対に格納する。 ただし、HLレジスタ対に与える除数が0000H(ゼロ)であった場合には、一定時間内蔵ブザー鳴動後、マシン後モニタの制御下に移る。 |
| 「MINUSO」 | DEレジスタ対に格納された16ビット長の整数の符号を反転する。すなわち、各ビットごとのNOTを求め、1を加える。 |
| 「RNDO」 | 1以上で、DEレジスタ対で指定する整数以下の乱数を発生し、DEレジスタ対に格納する。乱数の初期値はED52H~ED53H番地の内容を参照する。 |
| 「ABSO」 | DEレジスタ対に格納された16ビット長の整数の絶対値を求める。 具体的には、Dレジスタの最上位ビット(MSB)が1であれば2の補数を求めて、符号を反転する。 |
| 「INPUTO」 | キーボードから有符号16ビット長の整数を入力し、バイナリ形式に変換後、DEレジスタ対に格納する。 |
| 「OTCHRO」 | Aレジスタに格納されたキャラクタ・コードに従ったキャラクタを出力機器へ出力する。 ただし、EA58H番地の最上位ビット(MSB)が0のときCRT画面を、1のときCRT画面とプリンタを、それぞれ出力機器として選択する。 |
| 「OTNUMO」 | レジスタDEに格納された有符号の16ビット整数データを、10進数として周辺機器へ左づめ出力する。 ただし、EA58H番地の最上位ビット(MSB)が0のときCRT画面を、1のときCRT画面とプリンタを、それぞれ出力機器として選択する。 |



《第116図》2次元4目ならべ・汎用サブルーチン・パッケージ・アセンブル・リスト

```

;
;*****
;
;  subroutine package
;
;*****
;
002B PRNCHR:EQU 2BH
0035 DSPCHR:EQU 35H
0257 CONOUT:EQU 0257H
0F75 CONIN: EQU 0F75H
1B8A INPLN: EQU 1B8AH
4095 CPHLDE:EQU 4095H
5C66 MONHOT:EQU 5C66H
5E39 MONHCK:EQU 5E39H
5E4B MONHBN:EQU 5E4BH
5FB9 MONIN: EQU 5FB9H
5FC1 MONCAP:EQU 5FC1H
C500 TARGET:EQU 0C500H
EA58 TRMFLG:EQU 0EA58H
EC95 INBU: EQU 0EC95H
EC96 INBUFF:EQU 0EC96H
EC97 INBUF: EQU 0EC97H
ED00 BASE: EQU 0ED00H
ED4E DIVWK: EQU 0ED4EH
ED52 RNDBUF:EQU 0ED52H
ED68 DIVWK1:EQU 0ED68H
ED87 UBUF: EQU 0ED87H
ED88 UBUFF: EQU 0ED88H
FFFF BOTTOM:EQU 0FFFFH
;
;          ORG 0C100H
;
; jump table
;
C100 C340C3 RUN0: JP RUN
C103 C322C3 CHAR0: JP CHAR
C106 C351C1 TAB0: JP TAB
C109 C348C1 CRLF0: JP CRLF
C10C C35CC1 ESC0: JP ESC
C10F C364C1 JP ESC1
C112 C36BC1 INKEY0: JP INKEY
C115 C375C1 HFOUR0: JP HFOUR
C118 C374C1 HTW00: JP HTW0
C11B C390C1 USING0: JP USING
C11E C304C2 ISUB0: JP ISUB
C121 C332C2 IMULT0: JP IMULT
C124 C348C2 IDIV0: JP IDIV
C127 C36BC1 JP INKEY
C12A C32AC2 MINUS0: JP MINUS

```



```

C12D C37CC2  RND0:  JP  RND
C130 C39AC2  ABS0:  JP  ABS
C133 C3A0C2          JP  IF
C136 C3A5C2  INPUT0:JP  INPUT
C139 C371C1  OTCHR0:JP  OTCHR
C13C C326C3  OTNUM0:JP  OTNUM
C13F C3665C          JP  MONHOT
C142 C340C3          JP  RUN
C145 C340C3          JP  RUN

;
C148 3E0D    CRLF:  LD   A,0DH
C14A CD71C1          CALL OTCHR
C14D 3E0A          LD   A,0AH
C14F 1820          JR   OTCHR

;
; output spaces
;
C151 53      TAB:   LD   D,E
C152 14      TAB1:  INC  D
C153 15      TAB2:  DEC  D
C154 C8          RET  Z
C155 3E20          LD   A,' '
C157 CD71C1          CALL OTCHR
C15A 18F7          JR   TAB2

;
; check escape
;
C15C CD64C1  ESC:   CALL ESC1
C15F D0          RET  NC
C160 CDB95F          CALL MONIN
C163 C9          RET

;
C164 DB09      ESC1:  IN   A,(9)
C166 FE7F          CP   7FH
C168 C0          RET  NZ
C169 37          SCF
C16A C9          RET

;
; input a character
;
C16B CD750F  INKEY:  CALL CONIN
C16E CDC15F          CALL MONCAP
C171 C32CC3  OTCHR:  JP   OTCHR1

;
; output 2 digits hexa
;
C174 43      HTWO:   LD   B,E

;
; output 4 digits hexa
;
C175 78      HFOUR:  LD   A,B

```



```

C176 CD7EC1      CALL HSUB
C179 78          LD   A,B
C17A E60F        AND   0FH
C17C 1806        JR    HSUB1

;
C17E E6F0        HSUB:  AND   0F0H
C180 0F          RRCA
C181 0F          RRCA
C182 0F          RRCA
C183 0F          RRCA
C184 FE0A        HSUB1: CP    0AH
C186 FA8BC1      JP    M,HSUB2
C189 C607        ADD   A,7
C18B C630        HSUB2: ADD   A,'0'
C18D C371C1      JP    OTCHR

;
; output with format
;
C190 C5          USING: PUSH BC
C191 CDA3C1      CALL USUB
C194 C1          POP   BC
C195 7D          LD    A,L
C196 81          ADD   A,C
C197 D687        SUB   87H
C199 57          LD    D,A
C19A FA55C2      JP    M,OTMSG
C19D CD52C1      CALL TAB1
C1A0 C355C2      JP    OTMSG

;
C1A3 7A          USUB:  LD    A,D
C1A4 B7          OR     A
C1A5 FAABC1      JP    M,USUB1
C1A8 C3B5C1      JP    USUB2

;
C1AB CD2AC2      USUB1: CALL MINUS
C1AE CDB5C1      CALL USUB2
C1B1 2B          DEC   HL
C1B2 362D        LD    (HL),'-'
C1B4 C9          RET

;
C1B5 2187ED      USUB2: LD    HL,UBUF
C1B8 3600        LD    (HL),0
C1BA E5          USUB3: PUSH HL
C1BB 210A00      LD    HL,0AH
C1BE CDCDC1      CALL PACK
C1C1 7D          LD    A,L
C1C2 C630        ADD   A,'0'
C1C4 E1          POP   HL
C1C5 2B          DEC   HL
C1C6 77          LD    (HL),A
C1C7 7B          LD    A,E

```


| | | | |
|---------------|--------|------|----------|
| C1C8 B2 | | OR | D |
| C1C9 C2BAC1 | | JP | NZ,USUB3 |
| C1CC C9 | | RET | |
| ; | | | |
| C1CD 0E00 | PACK: | LD | C,0 |
| C1CF 0C | PACK1: | INC | C |
| C1D0 29 | | ADD | HL,HL |
| C1D1 D2CFC1 | | JP | NC,PACK1 |
| C1D4 CDFDC1 | | CALL | PACK6 |
| C1D7 E5 | | PUSH | HL |
| C1D8 210000 | | LD | HL,0 |
| C1DB E3 | | EX | (SP),HL |
| C1DC CD04C2 | PACK2: | CALL | ISUB |
| C1DF 3F | | CCF | |
| C1E0 DAE7C1 | | JP | C,PACK3 |
| C1E3 EB | | EX | DE,HL |
| C1E4 19 | | ADD | HL,DE |
| C1E5 EB | | EX | DE,HL |
| C1E6 B7 | | OR | A |
| C1E7 E3 | PACK3: | EX | (SP),HL |
| C1E8 7D | | LD | A,L |
| C1E9 17 | | RLA | |
| C1EA 6F | | LD | L,A |
| C1EB 7C | | LD | A,H |
| C1EC 17 | | RLA | |
| C1ED 67 | | LD | H,A |
| C1EE E3 | | EX | (SP),HL |
| C1EF 0D | | DEC | C |
| C1F0 CAF9C1 | | JP | Z,PACK4 |
| C1F3 CDFCC1 | | CALL | PACK5 |
| C1F6 C3DCC1 | | JP | PACK2 |
| ; | | | |
| C1F9 E1 | PACK4: | POP | HL |
| C1FA EB | | EX | DE,HL |
| C1FB C9 | | RET | |
| ; | | | |
| C1FC B7 | PACK5: | OR | A |
| C1FD 7C | PACK6: | LD | A,H |
| C1FE 1F | | RRA | |
| C1FF 67 | | LD | H,A |
| C200 7D | | LD | A,L |
| C201 1F | | RRA | |
| C202 6F | | LD | L,A |
| C203 C9 | | RET | |
| ; | | | |
| ; subtraction | | | |
| ; | | | |
| C204 7B | ISUB: | LD | A,E |
| C205 95 | | SUB | L |
| C206 5F | | LD | E,A |
| C207 7A | | LD | A,D |


```

C208 9C          SBC  A,H
C209 57          LD   D,A
C20A C9          RET

;
C20B F214C2      IDIVS: JP   P, IDIVS1
C20E 04          INC  B
C20F EB          EX   DE,HL
C210 CD2AC2      CALL MINUS
C213 EB          EX   DE,HL
C214 7A          IDIVS1: LD  A,D
C215 B7          OR   A
C216 F21DC2      JP   P, IDIVS2
C219 05          DEC  B
C21A CD2AC2      CALL MINUS
C21D CDCDC1      IDIVS2: CALL PACK
C220 224EED      LD   (DIVWK),HL
C223 2A68ED      LD   HL,(DIVWK1)
C226 3E00        LD   A,0
C228 B8          CP   B
C229 C8          RET  Z

;
; complement the sign
;
C22A 7A          MINUS: LD   A,D
C22B 2F          CPL
C22C 57          LD   D,A
C22D 7B          LD   A,E
C22E 2F          CPL
C22F 5F          LD   E,A
C230 13          INC  DE
C231 C9          RET

;
; multiplication
;
C232 D5          IMULT: PUSH DE
C233 0E10        LD   C,10H
C235 110000      LD   DE,0
C238 E3          IMULT1: EX  (SP),HL
C239 CDFCC1      CALL PACK5
C23C E3          EX  (SP),HL
C23D 3003        JR   NC, IMULT2
C23F EB          EX  DE,HL
C240 19          ADD  HL,DE
C241 EB          EX  DE,HL
C242 29          IMULT2: ADD HL,HL
C243 0D          DEC  C
C244 20F2        JR   NZ, IMULT1
C246 C1          POP  BC
C247 C9          RET

;
; division

```



```

;
C248 0600      IDIV:  LD    B,0
C24A 7C         LD    A,H
C24B B7         OR     A
C24C 20BD       JR     NZ,IDIVS
C24E B5         OR     L
C24F 20C3       JR     NZ,IDIVS1
C251 3E2F       LD    A,'/'
C253 180D       JR     DIVERR
;
; output message
;
C255 AF         OTMSG: XOR    A
C256 47         LD    B,A
C257 7E         OTMSG1:LD    A,(HL)
C258 23         INC    HL
C259 B8         CP     B
C25A CA5CC1     JP     Z,ESC
C25D CD71C1     CALL   OTCHR
C260 18F5       JR     OTMSG1
;
; error of division
;
C262 3E07       DIVERR:LD    A,7
C264 CD71C1     CALL   OTCHR
C267 C3665C     JP     MONHOT
;
C26A 200A       IF1:   JR     NZ,IF2
C26C 7B         LD    A,E
C26D B7         OR     A
C26E 2006       JR     NZ,IF2
C270 110100     LD    DE,1
C273 F6FF       OR     0FFH
C275 C9         RET
;
C276 110000     IF2:   LD    DE,0
C279 E600       AND    0
C27B C9         RET
;
; generate random number
;
C27C E5         RND:   PUSH   HL
C27D D5         PUSH   DE
C27E 2A52ED     LD     HL,(RNDBUF)
C281 11093D     LD     DE,3D09H
C284 CD32C2     CALL   IMULT
C287 13         INC    DE
C288 EB         EX     DE,HL
C289 2252ED     LD     (RNDBUF),HL
C28C EB         EX     DE,HL
C28D E1         POP    HL

```



```

C28E 5A          LD    E,D
C28F 1600        LD    D,0
C291 CD32C2      CALL  IMULT
C294 5A          LD    E,D
C295 1600        LD    D,0
C297 E1          POP   HL
C298 13          INC   DE
C299 C9          RET

;
; get absolute number
;
C29A 7A          ABS:  LD    A,D
C29B A7          AND    A
C29C FA2AC2      JP     M,MINUS
C29F C9          RET

;
; check de=zero
;
C2A0 7A          IF:   LD    A,D
C2A1 A7          AND    A
C2A2 C36AC2      JP     IF1

;
; input a number
;
C2A5 3E3F        INPUT: LD    A,'?'
C2A7 CD5702      CALL  CONOUT
C2AA CD8A1B      CALL  INPLN
C2AD 2196EC      LD    HL,INBUFF
C2B0 7E          LD    A,(HL)
C2B1 FE2B        CP     '+'
C2B3 280F        JR     Z,INPUT1
C2B5 FE2D        CP     '-'
C2B7 2812        JR     Z,INPUT2
C2B9 FE24        CP     '$'
C2BB 281B        JR     Z,INPUT3
C2BD 2195EC      LD    HL,INBU
C2C0 CDFDC2      CALL  INPUT7
C2C3 C9          RET

;
INPUT1: LD    HL,INBUFF
C2C4 2196EC      CALL  INPUT7
C2C7 CDFDC2
C2CA C9          RET

;
INPUT2: LD    HL,INBUFF
C2CB 2196EC      CALL  INPUT7
C2CE CDFDC2
C2D1 210000      LD    HL,0
C2D4 ED52        SBC   HL,DE
C2D6 EB          EX    DE,HL
C2D7 C9          RET

;
C2D8 1197EC      INPUT3: LD    DE,INBUF

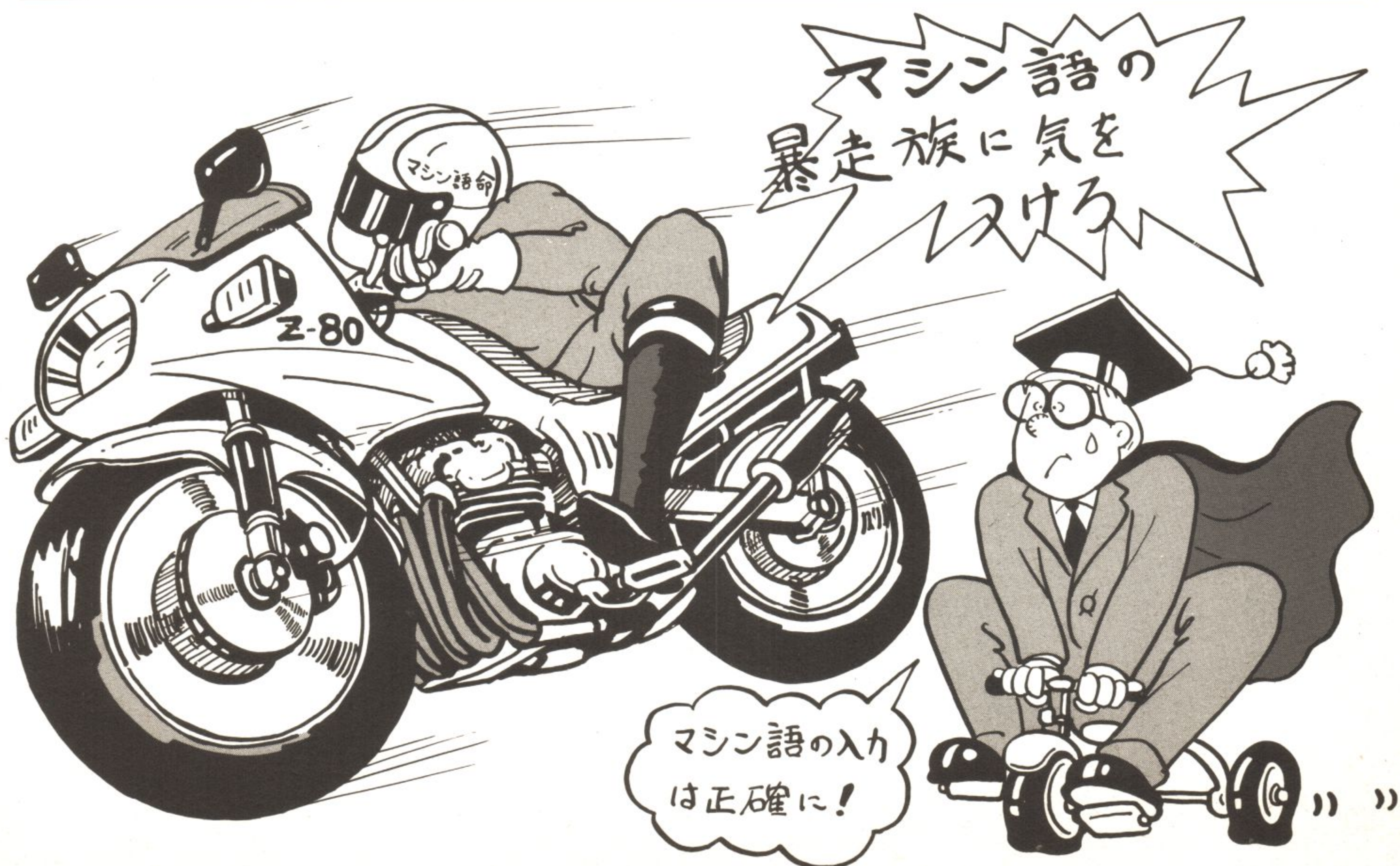
```


| | | | |
|----------------------|--------|------------|----------|
| C2DB | 210000 | LD | HL,0 |
| C2DE | 1A | INPUT4:LD | A,(DE) |
| C2DF | CDC15F | CALL | MONCAP |
| C2E2 | FE00 | CP | 0 |
| C2E4 | 280B | JR | Z,INPUT5 |
| C2E6 | CD395E | CALL | MONHCK |
| C2E9 | 3815 | JR | C,INPUT8 |
| C2EB | CD4B5E | CALL | MONHBN |
| C2EE | 13 | INC | DE |
| C2EF | 18ED | JR | INPUT4 |
| ; | | | |
| C2F1 | EB | INPUT5:EX | DE,HL |
| C2F2 | C9 | RET | |
| ; | | | |
| C2F3 | E1 | INPUT6:POP | HL |
| C2F4 | 37 | SCF | |
| C2F5 | 3F | CCF | |
| C2F6 | 3E07 | LD | A,7 |
| C2F8 | CD5702 | CALL | CONOUT |
| C2FB | 18A8 | JR | INPUT |
| ; | | | |
| C2FD | 110000 | INPUT7:LD | DE,0 |
| C300 | D7 | INPUT8:RST | 10H |
| C301 | D0 | RET | NC |
| C302 | E5 | PUSH | HL |
| C303 | F5 | PUSH | AF |
| C304 | 218819 | LD | HL,1988H |
| C307 | CD9540 | CALL | CPHLDE |
| C30A | DA1EC3 | JP | C,INPUT9 |
| C30D | 62 | LD | H,D |
| C30E | 6B | LD | L,E |
| C30F | 19 | ADD | HL,DE |
| C310 | 29 | ADD | HL,HL |
| C311 | 19 | ADD | HL,DE |
| C312 | 29 | ADD | HL,HL |
| C313 | F1 | POP | AF |
| C314 | D630 | SUB | '0' |
| C316 | 5F | LD | E,A |
| C317 | 1600 | LD | D,0 |
| C319 | 19 | ADD | HL,DE |
| C31A | EB | EX | DE,HL |
| C31B | E1 | POP | HL |
| C31C | 18E2 | JR | INPUT8 |
| ; | | | |
| C31E | F1 | INPUT9:POP | AF |
| C31F | E1 | POP | HL |
| C320 | 18D1 | JR | INPUT6 |
| ; | | | |
| ; output a character | | | |
| ; | | | |
| C322 | 7B | CHAR: LD | A,E |


```

C323 C371C1          JP   OTCHR
;
; output a number
;
C326 CDA3C1          OTNUM: CALL USUB
C329 C355C2          JP   OTMSG
;
C32C F5              OTCHR1: PUSH AF
C32D 3A58EA          LD    A, (TRMFLG)
C330 FE09            CP    9
C332 2805            JR    Z, OTCHR3
C334 F1              POP   AF
C335 CD3500          OTCHR2: CALL DSPCHR
C338 C9              RET
;
C339 F1              OTCHR3: POP   AF
C33A CD2B00          CALL PRNCHR
C33D 18F6            JR    OTCHR2
;
C33F 00              NOP
;
; execute the target
;
C340 31FFFF          RUN:   LD    SP, BOTTOM
C343 2100ED          LD    HL, BASE
C346 2288ED          LD    (UBUFF), HL
C349 C300C5          JP    TARGET
;
C34C                  END

```



サブルーチン・アセンブル・リスト

《第117図》 2次元4目ならべ・サブルーチン・アセンブル・リスト

```

;
;*****
;
; 2d4m subroutines
;
;*****
;
C11E      ISUB0: EQU    0C11EH
C121      IMULT0:EQU    0C121H
C124      IDIVO: EQU    0C124H
C12A      MINUS0:EQU    0C12AH
C139      OTCHRO:EQU    0C139H
CC94      MAIN:  EQU    0CC94H
ED06      STONE: EQU    0ED06H
ED0C      STONE1:EQU    0ED0CH
ED0E      WORK1: EQU    0ED0EH
ED10      COORD1:EQU    0ED10H
ED12      COORD2:EQU    0ED12H
ED14      WORK2: EQU    0ED14H
ED16      DEEP:  EQU    0ED16H
ED18      DEEP1: EQU    0ED18H
ED1A      WORK3: EQU    0ED1AH
ED1C      WORK4: EQU    0ED1CH
ED1E      VECT1: EQU    0ED1EH
ED20      VECT2: EQU    0ED20H
ED22      SPACES:EQU    0ED22H
ED24      COORD3:EQU    0ED24H
ED26      COORD4:EQU    0ED26H
ED28      VALUE: EQU    0ED28H
ED2C      ABFLAG:EQU    0ED2CH
ED2E      PATR:  EQU    0ED2EH
ED30      VECTOR:EQU    0ED30H

```


| | | | |
|-------------|-----------------|------------|---|
| ED32 | STACK: EQU | 0ED32H | |
| ED34 | COORD: EQU | 0ED34H | |
| ED36 | PCUS: EQU | 0ED36H | |
| ED38 | BOARD: EQU | 0ED38H | |
| ED4E | DIVWK: EQU | 0ED4EH | |
| | ; | | |
| | ORG | 0C500H | |
| | ; | | |
| | ; entry of 2d4m | | |
| | ; | | |
| C500 C394CC | TARGET:JP | MAIN | } メインルーチンへジャンプする |
| | ; | | |
| | ; next stone | | |
| | ; | | |
| C503 210000 | LIST: LD | HL,0 | } (WORK1+1) (WORK1) を0に クリアする |
| C506 220EED | LD | (WORK1),HL | |
| C509 210000 | LD | HL,0 | } (WORK2+1) (WORK2) をルー プ・カウンタとして0から5までくり返 す |
| C50C 2214ED | LD | (WORK2),HL | |
| C50F 210500 | LD | HL,5 | |
| C512 EB | EX | DE,HL | |
| C513 CD16C5 | CALL | LIST1 | } (BOARD+1) (BOARD) の内容 に(WORK2+1) (WORK2) の内 容を加えて指定するアドレスの内容が0 を越えている場合には『LIST6』に ジャンプする |
| C516 D5 | LIST1: PUSH | DE | |
| C517 D5 | PUSH | DE | |
| C518 2A14ED | LD | HL,(WORK2) | |
| C51B EB | EX | DE,HL | |
| C51C 2A38ED | LD | HL,(BOARD) | |
| C51F 19 | ADD | HL,DE | |
| C520 D1 | POP | DE | |
| C521 6E | LD | L,(HL) | |
| C522 2600 | LD | H,0 | |
| C524 EB | EX | DE,HL | |
| C525 210000 | LD | HL,0 | |
| C528 EB | EX | DE,HL | |
| C529 CD1EC1 | CALL | ISUB0 | |
| C52C F232C5 | JP | P,LIST2 | |
| C52F C3A2C5 | JP | LIST6 | |
| | ; | | |
| C532 210600 | LIST2: LD | HL,6 | } (WORK3+1) (WORK3) に00 06Hを格納する |
| C535 221AED | LD | (WORK3),HL | |
| C538 D5 | LIST3: PUSH | DE | } (BOARD+1) (BOARD) の内容に (WORK2+1) (WORK2) の内容 を加えさらに(WORK3+1) (WOR K3) の内容を加えて指定するアドレ スの内容が0を越えている場合には『LI ST5』にジャンプする |
| C539 2A14ED | LD | HL,(WORK2) | |
| C53C EB | EX | DE,HL | |
| C53D 2A1AED | LD | HL,(WORK3) | |
| C540 19 | ADD | HL,DE | |
| C541 EB | EX | DE,HL | |
| C542 2A38ED | LD | HL,(BOARD) | |
| C545 19 | ADD | HL,DE | |
| C546 D1 | POP | DE | |
| C547 6E | LD | L,(HL) | |
| C548 2600 | LD | H,0 | |
| C54A EB | EX | DE,HL | |

| | | | | | |
|------|--------|--------|------|------------|---|
| C54B | 210000 | | LD | HL,0 | |
| C54E | EB | | EX | DE,HL | |
| C54F | CD1EC1 | | CALL | ISUB0 | |
| C552 | F258C5 | | JP | P,LIST4 | |
| C555 | C373C5 | | JP | LIST5 | |
| ; | | | | | |
| C558 | 2A1AED | LIST4: | LD | HL,(WORK3) | |
| C55B | EB | | EX | DE,HL | |
| C55C | 210600 | | LD | HL,6 | (WORK3+1)(WORK3)に0006Hを加える |
| C55F | 19 | | ADD | HL,DE | |
| C560 | 221AED | | LD | (WORK3),HL | |
| C563 | 2A1AED | | LD | HL,(WORK3) | |
| C566 | EB | | EX | DE,HL | |
| C567 | 212400 | | LD | HL,24H | (WORK3+1)(WORK3)が36未満のとき『LIST3』にもどってくり返す |
| C56A | CD1EC1 | | CALL | ISUB0 | |
| C56D | F273C5 | | JP | P,LIST5 | |
| C570 | C338C5 | | JP | LIST3 | |
| ; | | | | | |
| C573 | 2A16ED | LIST5: | LD | HL,(DEEP) | |
| C576 | EB | | EX | DE,HL | |
| C577 | 210200 | | LD | HL,2 | |
| C57A | 19 | | ADD | HL,DE | |
| C57B | EB | | EX | DE,HL | |
| C57C | 2A0EED | | LD | HL,(WORK1) | |
| C57F | 19 | | ADD | HL,DE | |
| C580 | EB | | EX | DE,HL | |
| C581 | 2A32ED | | LD | HL,(STACK) | (STACK+1)(STACK)の内容に(WORK1+1)(WORK1)の内容を加え2を加えさらに(DEEP+1)(DEEP)を加えて指定するアドレスに(WORK2+1)(WORK2)の内容に(WORK3+1)(WORK3)の内容を加えさらに6を減じたものを格納する |
| C584 | 19 | | ADD | HL,DE | |
| C585 | E5 | | PUSH | HL | |
| C586 | 2A14ED | | LD | HL,(WORK2) | |
| C589 | EB | | EX | DE,HL | |
| C58A | 2A1AED | | LD | HL,(WORK3) | |
| C58D | 19 | | ADD | HL,DE | |
| C58E | EB | | EX | DE,HL | |
| C58F | 210600 | | LD | HL,6 | |
| C592 | CD1EC1 | | CALL | ISUB0 | |
| C595 | E1 | | POP | HL | |
| C596 | 73 | | LD | (HL),E | |
| C597 | 2A0EED | | LD | HL,(WORK1) | |
| C59A | EB | | EX | DE,HL | |
| C59B | 210100 | | LD | HL,1 | (WORK1+1)(WORK1)に1を格納する |
| C59E | 19 | | ADD | HL,DE | |
| C59F | 220EED | | LD | (WORK1),HL | |
| C5A2 | 2A14ED | LIST6: | LD | HL,(WORK2) | |
| C5A5 | EB | | EX | DE,HL | |
| C5A6 | 210100 | | LD | HL,1 | |
| C5A9 | 19 | | ADD | HL,DE | |
| C5AA | 2214ED | | LD | (WORK2),HL | |
| C5AD | D1 | | POP | DE | |
| C5AE | D5 | | PUSH | DE | |
| C5AF | CD1EC1 | | CALL | ISUB0 | (WORK2+1)(WORK2)をループ・カウンタとするくり返し処理の終了をチェックする |

| | | | | |
|-------------------|--------|--------|---------|-------------|
| C5B2 | D1 | POP | DE | |
| C5B3 | E1 | POP | HL | |
| C5B4 | FAB9C5 | JP | M,LIST7 | |
| C5B7 | E5 | PUSH | HL | |
| C5B8 | E9 | JP | (HL) | |
| ; | | | | |
| C5B9 | 2A16ED | LIST7: | LD | HL,(DEEP) |
| C5BC | EB | | EX | DE,HL |
| C5BD | 2A32ED | | LD | HL,(STACK) |
| C5C0 | 19 | | ADD | HL,DE |
| C5C1 | E5 | | PUSH | HL |
| C5C2 | 2A0EED | | LD | HL,(WORK1) |
| C5C5 | EB | | EX | DE,HL |
| C5C6 | 210200 | | LD | HL,2 |
| C5C9 | 19 | | ADD | HL,DE |
| C5CA | EB | | EX | DE,HL |
| C5CB | E1 | | POP | HL |
| C5CC | 73 | | LD | (HL),E |
| C5CD | C9 | | RET | |
| ; | | | | |
| ; calculate value | | | | |
| ; | | | | |
| C5CE | 210000 | VAL: | LD | HL,0 |
| C5D1 | 2228ED | | LD | (VALUE),HL |
| C5D4 | 210000 | | LD | HL,0 |
| C5D7 | 221AED | | LD | (WORK3),HL |
| C5DA | D5 | VAL1: | PUSH | DE |
| C5DB | 2A1AED | | LD | HL,(WORK3) |
| C5DE | EB | | EX | DE,HL |
| C5DF | 2A30ED | | LD | HL,(VECTOR) |
| C5E2 | 19 | | ADD | HL,DE |
| C5E3 | D1 | | POP | DE |
| C5E4 | 6E | | LD | L,(HL) |
| C5E5 | 2600 | | LD | H,0 |
| C5E7 | 221EED | | LD | (VECT1),HL |
| C5EA | D5 | | PUSH | DE |
| C5EB | 2A1AED | | LD | HL,(WORK3) |
| C5EE | EB | | EX | DE,HL |
| C5EF | 210100 | | LD | HL,1 |
| C5F2 | 19 | | ADD | HL,DE |
| C5F3 | EB | | EX | DE,HL |
| C5F4 | 2A30ED | | LD | HL,(VECTOR) |
| C5F7 | 19 | | ADD | HL,DE |
| C5F8 | D1 | | POP | DE |
| C5F9 | 6E | | LD | L,(HL) |
| C5FA | 2600 | | LD | H,0 |
| C5FC | 2220ED | | LD | (VECT2),HL |
| C5FF | 2A1EED | | LD | HL,(VECT1) |
| C602 | EB | | EX | DE,HL |
| C603 | 21FF00 | | LD | HL,OFFH |
| C606 | CD1EC1 | | CALL | ISUB0 |

(STACK+1) (STACK) の内容に (DEEP+1) (DEEP) の内容を加えて指定するアドレスに (WORK1+1) (WORK1) の内容に2を加えたものを格納する

メインルーチンへもどる

(VALUE+1) (VALUE) を0にクリアする

(WORK3+1) (WORK3) を0にクリアする

(VECTOR+1) (VECTOR) の内容に (WORK3+1) (WORK3) の内容を加えて指定するアドレスの内容を (VECT1+1) (VECT1) に格納する

(VECTOR+1) (VECTOR) の内容に (WORK3+1) (WORK3) の内容を加えさらに1を加えて指定するアドレスの内容を (VECT2+1) (VECT2) に格納する

| | | | | | |
|------|--------|-------|------|--------------|--|
| C609 | B3 | | OR | E | |
| C60A | C21AC6 | | JP | NZ, VAL2 | |
| C60D | D5 | | PUSH | DE | (VECT1+1) (VECT1) の内容 が FFH であれば -1 に変換する |
| C60E | 210100 | | LD | HL, 1 | |
| C611 | EB | | EX | DE, HL | |
| C612 | CD2AC1 | | CALL | MINUS0 | |
| C615 | EB | | EX | DE, HL | |
| C616 | D1 | | POP | DE | |
| C617 | 221EED | | LD | (VECT1), HL | |
| C61A | 2A10ED | VAL2: | LD | HL, (COORD1) | (COORD1+1) (COORD1) の内 容を (COORD3+1) (COORD3) に格納する |
| C61D | 2224ED | | LD | (COORD3), HL | |
| C620 | 2A12ED | | LD | HL, (COORD2) | (COORD2+1) (COORD2) の 内容を (COORD4+1) (COORD 4) に格納する |
| C623 | 2226ED | | LD | (COORD4), HL | |
| C626 | 210100 | | LD | HL, 1 | (WORK1+1) (WORK1) に 1 を格 納する |
| C629 | 220EED | | LD | (WORK1), HL | |
| C62C | 210000 | | LD | HL, 0 | (SPACES+1) (SPACES) に 0 を格納する |
| C62F | 2222ED | | LD | (SPACES), HL | |
| C632 | 2A24ED | VAL3: | LD | HL, (COORD3) | |
| C635 | EB | | EX | DE, HL | (COORD3+1) (COORD3) に (VECT1+1) (VECT1) の内容 を加える |
| C636 | 2A1EED | | LD | HL, (VECT1) | |
| C639 | 19 | | ADD | HL, DE | |
| C63A | 2224ED | | LD | (COORD3), HL | |
| C63D | 2A26ED | | LD | HL, (COORD4) | |
| C640 | EB | | EX | DE, HL | (COORD4+1) (COORD4) に (VECT2+1) (VECT2) の内容 を加える |
| C641 | 2A20ED | | LD | HL, (VECT2) | |
| C644 | 19 | | ADD | HL, DE | |
| C645 | 2226ED | | LD | (COORD4), HL | |
| C648 | D5 | | PUSH | DE | |
| C649 | 2A24ED | | LD | HL, (COORD3) | |
| C64C | EB | | EX | DE, HL | |
| C64D | 210500 | | LD | HL, 5 | |
| C650 | EB | | EX | DE, HL | |
| C651 | CD1EC1 | | CALL | ISUB0 | |
| C654 | 110000 | | LD | DE, 0 | |
| C657 | F25BC6 | | JP | P, VAL4 | |
| C65A | 1C | | INC | E | |
| C65B | EB | VAL4: | EX | DE, HL | |
| C65C | D1 | | POP | DE | |
| C65D | EB | | EX | DE, HL | |
| C65E | D5 | | PUSH | DE | |
| C65F | 2A24ED | | LD | HL, (COORD3) | |
| C662 | EB | | EX | DE, HL | |
| C663 | 210000 | | LD | HL, 0 | |
| C666 | CD1EC1 | | CALL | ISUB0 | |
| C669 | 110000 | | LD | DE, 0 | |
| C66C | F270C6 | | JP | P, VAL5 | |
| C66F | 1C | | INC | E | (COORD3+1) (COORD3) の 内容が 0 ~ 5 の範囲にないかまたは (C OORD4+1) (COORD4) の内容 が 5 を越える場合には『VAL9』ヘジ ャンプする |
| C670 | EB | VAL5: | EX | DE, HL | |
| C671 | D1 | | POP | DE | |
| C672 | 19 | | ADD | HL, DE | |
| C673 | EB | | EX | DE, HL | |

| | | | | | |
|------|--------|---------|------|--------------|---|
| C674 | D5 | | PUSH | DE | |
| C675 | 2A26ED | | LD | HL, (COORD4) | |
| C678 | EB | | EX | DE, HL | |
| C679 | 210500 | | LD | HL, 5 | |
| C67C | EB | | EX | DE, HL | |
| C67D | CD1EC1 | | CALL | ISUB0 | |
| C680 | 110000 | | LD | DE, 0 | |
| C683 | F287C6 | | JP | P, VAL6 | |
| C686 | 1C | | INC | E | |
| C687 | EB | VAL6: | EX | DE, HL | |
| C688 | D1 | | POP | DE | |
| C689 | 19 | | ADD | HL, DE | |
| C68A | EB | | EX | DE, HL | |
| C68B | 7B | | LD | A, E | |
| C68C | B2 | | OR | D | |
| C68D | CA93C6 | | JP | Z, VAL7 | |
| C690 | C3E3C6 | | JP | VAL9 | |
| | | | | | |
| C693 | D5 | ; VAL7: | PUSH | DE | |
| C694 | 2A26ED | | LD | HL, (COORD4) | |
| C697 | EB | | EX | DE, HL | |
| C698 | 210600 | | LD | HL, 6 | |
| C69B | CD21C1 | | CALL | IMULT0 | |
| C69E | 2A24ED | | LD | HL, (COORD3) | (BOARD+1) (BOARD) の内容に (COORD4+1) (COORD4) の内容を6倍して (COORD3+1) (COORD3) の内容を加えたものを加えて指定するアドレスの内容を (WORK2+1) (WORK2) に格納する |
| C6A1 | 19 | | ADD | HL, DE | |
| C6A2 | EB | | EX | DE, HL | |
| C6A3 | 2A38ED | | LD | HL, (BOARD) | |
| C6A6 | 19 | | ADD | HL, DE | |
| C6A7 | D1 | | POP | DE | |
| C6A8 | 6E | | LD | L, (HL) | |
| C6A9 | 2600 | | LD | H, 0 | |
| C6AB | 2214ED | | LD | (WORK2), HL | |
| C6AE | 2A14ED | | LD | HL, (WORK2) | |
| C6B1 | EB | | EX | DE, HL | |
| C6B2 | 2A0CED | | LD | HL, (STONE1) | |
| C6B5 | CD1EC1 | | CALL | ISUB0 | |
| C6B8 | B3 | | OR | E | |
| C6B9 | C2CAC6 | | JP | NZ, VAL8 | (WORK2+1) (WORK2) の内容と (STONE1+1) (STONE1) の内容が等しい場合には (WORK1+1) (WORK1) の内容に1を加えて『VAL3』にもどる |
| C6BC | 2A0EED | | LD | HL, (WORK1) | |
| C6BF | EB | | EX | DE, HL | |
| C6C0 | 210100 | | LD | HL, 1 | |
| C6C3 | 19 | | ADD | HL, DE | |
| C6C4 | 220EED | | LD | (WORK1), HL | |
| C6C7 | C332C6 | | JP | VAL3 | |
| | | | | | |
| C6CA | 2A14ED | ; VAL8: | LD | HL, (WORK2) | |
| C6CD | EB | | EX | DE, HL | |
| C6CE | 210000 | | LD | HL, 0 | |
| C6D1 | CD1EC1 | | CALL | ISUB0 | |
| C6D4 | B3 | | OR | E | |
| C6D5 | C2E3C6 | | JP | NZ, VAL9 | (WORK2+1) (WORK2) の内容が0である場合には (SPACES+1) (SPACES) の内容に1を加える |

| | | | | | |
|------|--------|--------|------|-------------|--|
| C6D8 | 2A22ED | | LD | HL,(SPACES) | |
| C6DB | EB | | EX | DE,HL | |
| C6DC | 210100 | | LD | HL,1 | |
| C6DF | 19 | | ADD | HL,DE | |
| C6E0 | 2222ED | | LD | (SPACES),HL | |
| C6E3 | 2A10ED | VAL9: | LD | HL,(COORD1) | (COORD1+1)(COORD1)の内容を(COORD3+1)(COORD3)に格納する |
| C6E6 | 2224ED | | LD | (COORD3),HL | |
| C6E9 | 2A12ED | | LD | HL,(COORD2) | (COORD2+1)(COORD2)の内容を(COORD4+1)(COORD4)に格納する |
| C6EC | 2226ED | | LD | (COORD4),HL | |
| C6EF | 2A24ED | VAL10: | LD | HL,(COORD3) | |
| C6F2 | EB | | EX | DE,HL | |
| C6F3 | 2A1EED | | LD | HL,(VECT1) | (COORD3+1)(COORD3)の内容から(VECT1+1)(VECT1)の内容を減じる |
| C6F6 | CD1EC1 | | CALL | ISUB0 | |
| C6F9 | EB | | EX | DE,HL | |
| C6FA | 2224ED | | LD | (COORD3),HL | |
| C6FD | 2A26ED | | LD | HL,(COORD4) | |
| C700 | EB | | EX | DE,HL | |
| C701 | 2A20ED | | LD | HL,(VECT2) | (COORD4+1)(COORD4)の内容から(VECT2+1)(VECT2)の内容を減じる |
| C704 | CD1EC1 | | CALL | ISUB0 | |
| C707 | EB | | EX | DE,HL | |
| C708 | 2226ED | | LD | (COORD4),HL | |
| C70B | D5 | | PUSH | DE | |
| C70C | 2A24ED | | LD | HL,(COORD3) | |
| C70F | EB | | EX | DE,HL | |
| C710 | 210500 | | LD | HL,5 | |
| C713 | EB | | EX | DE,HL | |
| C714 | CD1EC1 | | CALL | ISUB0 | |
| C717 | 110000 | | LD | DE,0 | |
| C71A | F21EC7 | | JP | P,VAL11 | |
| C71D | 1C | | INC | E | |
| C71E | EB | VAL11: | EX | DE,HL | |
| C71F | D1 | | POP | DE | |
| C720 | EB | | EX | DE,HL | |
| C721 | D5 | | PUSH | DE | |
| C722 | 2A24ED | | LD | HL,(COORD3) | |
| C725 | EB | | EX | DE,HL | |
| C726 | 210000 | | LD | HL,0 | |
| C729 | CD1EC1 | | CALL | ISUB0 | |
| C72C | 110000 | | LD | DE,0 | |
| C72F | F233C7 | | JP | P,VAL12 | |
| C732 | 1C | | INC | E | |
| C733 | EB | VAL12: | EX | DE,HL | (COORD3+1)(COORD3)の内容が0~5の範囲にないかまたは(COORD4+1)・(COORD4)の内容が0未満の場合には『VAL16』へジャンプする |
| C734 | D1 | | POP | DE | |
| C735 | 19 | | ADD | HL,DE | |
| C736 | EB | | EX | DE,HL | |
| C737 | D5 | | PUSH | DE | |
| C738 | 2A26ED | | LD | HL,(COORD4) | |
| C73B | EB | | EX | DE,HL | |
| C73C | 210000 | | LD | HL,0 | |
| C73F | CD1EC1 | | CALL | ISUB0 | |
| C742 | 110000 | | LD | DE,0 | |

| | | | | |
|------|--------|-------------|-------------|----------------------|
| C745 | F249C7 | JP | P,VAL13 | |
| C748 | 1C | INC | E | |
| C749 | EB | VAL13: EX | DE,HL | |
| C74A | D1 | POP | DE | |
| C74B | 19 | ADD | HL,DE | |
| C74C | EB | EX | DE,HL | |
| C74D | 7B | LD | A,E | |
| C74E | B2 | OR | D | |
| C74F | CA55C7 | JP | Z,VAL14 | |
| C752 | C3A5C7 | JP | VAL16 | |
| ; | | | | |
| C755 | D5 | VAL14: PUSH | DE | |
| C756 | 2A26ED | LD | HL,(COORD4) | |
| C759 | EB | EX | DE,HL | |
| C75A | 210600 | LD | HL,6 | |
| C75D | CD21C1 | CALL | IMULT0 | (BOARD+1)(BOARD)の内容 |
| C760 | 2A24ED | LD | HL,(COORD3) | に(COORD4+1)(COORD4) |
| C763 | 19 | ADD | HL,DE | の内容を6倍して(COORD3+1) |
| C764 | EB | EX | DE,HL | (COORD3)の内容を加えたものを加 |
| C765 | 2A38ED | LD | HL,(BOARD) | えて指定するアドレスの内容を(WOR |
| C768 | 19 | ADD | HL,DE | K2+1)(WORK2)に格納する |
| C769 | D1 | POP | DE | |
| C76A | 6E | LD | L,(HL) | |
| C76B | 2600 | LD | H,0 | |
| C76D | 2214ED | LD | (WORK2),HL | |
| C770 | 2A14ED | LD | HL,(WORK2) | |
| C773 | EB | EX | DE,HL | |
| C774 | 2A0CED | LD | HL,(STONE1) | |
| C777 | CD1EC1 | CALL | ISUB0 | (WORK2+1)(WORK2)の内容 |
| C77A | B3 | OR | E | と(STONE1+1)(STONE1)の |
| C77B | C28CC7 | JP | NZ,VAL15 | 内容が等しい場合には(WORK1+1) |
| C77E | 2A0EED | LD | HL,(WORK1) | (WORK1)の内容に1を加えて『VAL |
| C781 | EB | EX | DE,HL | 10』にもどる |
| C782 | 210100 | LD | HL,1 | |
| C785 | 19 | ADD | HL,DE | |
| C786 | 220EED | LD | (WORK1),HL | |
| C789 | C3EFC6 | JP | VAL10 | |
| ; | | | | |
| C78C | 2A14ED | VAL15: LD | HL,(WORK2) | |
| C78F | EB | EX | DE,HL | |
| C790 | 210000 | LD | HL,0 | |
| C793 | CD1EC1 | CALL | ISUB0 | |
| C796 | B3 | OR | E | |
| C797 | C2A5C7 | JP | NZ,VAL16 | (WORK2+1)(WORK2)の内容 |
| C79A | 2A22ED | LD | HL,(SPACES) | が0の場合には(SPACES+1)(S |
| C79D | EB | EX | DE,HL | PACES)の内容に1を加える |
| C79E | 210100 | LD | HL,1 | |
| C7A1 | 19 | ADD | HL,DE | |
| C7A2 | 2222ED | LD | (SPACES),HL | |
| C7A5 | 2A0EED | VAL16: LD | HL,(WORK1) | |
| C7A8 | EB | EX | DE,HL | |

| | | | | | |
|--------------|--------|------------|-------------|---|--|
| C7A9 | 210300 | LD | HL,3 | } | (WORK1+1) (WORK1) の内容 が3を超える場合には(VALUE+1) (VALUE)に格納して『VAL18』 へジャンプする |
| C7AC | EB | EX | DE,HL | | |
| C7AD | CD1EC1 | CALL | ISUB0 | | |
| C7B0 | F2BCC7 | JP | P,VAL17 | | |
| C7B3 | 21C800 | LD | HL,0C8H | | |
| C7B6 | 2228ED | LD | (VALUE),HL | | |
| C7B9 | C3F3C7 | JP | VAL18 | | |
| ; | | | | | |
| C7BC | 2A0EED | VAL17: LD | HL,(WORK1) | } | (WORK1+1) (WORK1) の内容 を2乗して(SPACES+1) (SPA CES)の内容を乗じ1を加えさらに(V ALUE+1) (VALUE)の内容を加 えたものを(VALUE+1) (VALU E) に格納する |
| C7BF | EB | EX | DE,HL | | |
| C7C0 | 2A0EED | LD | HL,(WORK1) | | |
| C7C3 | CD21C1 | CALL | IMULT0 | | |
| C7C6 | 2A22ED | LD | HL,(SPACES) | | |
| C7C9 | CD21C1 | CALL | IMULT0 | | |
| C7CC | 210100 | LD | HL,1 | } | (WORK3+1) (WORK3) の内容 に2を加える |
| C7CF | 19 | ADD | HL,DE | | |
| C7D0 | EB | EX | DE,HL | | |
| C7D1 | 2A28ED | LD | HL,(VALUE) | | |
| C7D4 | 19 | ADD | HL,DE | | |
| C7D5 | 2228ED | LD | (VALUE),HL | | |
| C7D8 | 2A1AED | LD | HL,(WORK3) | } | (WORK3+1) (WORK3) の内容 が7未満の場合には『VAL1』へもど ってきて返す |
| C7DB | EB | EX | DE,HL | | |
| C7DC | 210200 | LD | HL,2 | | |
| C7DF | 19 | ADD | HL,DE | | |
| C7E0 | 221AED | LD | (WORK3),HL | | |
| C7E3 | 2A1AED | LD | HL,(WORK3) | | |
| C7E6 | EB | EX | DE,HL | } | (STONE+1) (STONE) の内容 と(STONE1+1) (STONE1) の内容が異なる場合には201から (V ALUE+1) (VALUE)の内容を減 じて(VALUE+1) (VALUE) に 格納する |
| C7E7 | 210700 | LD | HL,7 | | |
| C7EA | CD1EC1 | CALL | ISUB0 | | |
| C7ED | F2F3C7 | JP | P,VAL18 | | |
| C7F0 | C3DAC5 | JP | VAL1 | | |
| ; | | | | | |
| C7F3 | 2A06ED | VAL18: LD | HL,(STONE) | } | (STONE+1) (STONE) の内容 と(STONE1+1) (STONE1) の内容が異なる場合には201から (V ALUE+1) (VALUE)の内容を減 じて(VALUE+1) (VALUE) に 格納する |
| C7F6 | EB | EX | DE,HL | | |
| C7F7 | 2A0CED | LD | HL,(STONE1) | | |
| C7FA | CD1EC1 | CALL | ISUB0 | | |
| C7FD | B3 | OR | E | | |
| C7FE | CA0FC8 | JP | Z,VAL19 | | |
| C801 | 21C900 | LD | HL,0C9H | } | メインルーチンへもどる |
| C804 | EB | EX | DE,HL | | |
| C805 | 2A28ED | LD | HL,(VALUE) | | |
| C808 | CD1EC1 | CALL | ISUB0 | | |
| C80B | EB | EX | DE,HL | | |
| C80C | 2228ED | LD | (VALUE),HL | | |
| C80F | C9 | VAL19: RET | | | |
| ; | | | | | |
| ; backup a-b | | | | | |
| ; | | | | | |
| C810 | 2A16ED | BACK: LD | HL,(DEEP) | } | (DEEP+1)(DEEP)の内容を (D EEP1+1)(DEEP1)に退避する |
| C813 | 2218ED | LD | (DEEP1),HL | | |
| C816 | 2A16ED | LD | HL,(DEEP) | | |

| | | | | |
|------|--------|--------|-------------|---|
| C819 | EB | EX | DE,HL | (DEEP+1)(DEEP)の内容から 8を減じる |
| C81A | 210800 | LD | HL,8 | |
| C81D | CD1EC1 | CALL | ISUB0 | |
| C820 | EB | EX | DE,HL | (STONE1+1)(STONE)の内 容が1であれば2に2であれば1に変換 する |
| C821 | 2216ED | LD | (DEEP),HL | |
| C824 | 210300 | LD | HL,3 | |
| C827 | EB | EX | DE,HL | (ABFLAG+1)(ABFLAG)に 0を格納する |
| C828 | 2A0CED | LD | HL,(STONE1) | |
| C82B | CD1EC1 | CALL | ISUB0 | |
| C82E | EB | EX | DE,HL | (DEEP+1)(DEEP)の内容を16 で割った剰余が8以上の場合には『BA CK3』へジャンプする |
| C82F | 220CED | LD | (STONE1),HL | |
| C832 | 210000 | LD | HL,0 | |
| C835 | 222CED | LD | (ABFLAG),HL | |
| C838 | D5 | PUSH | DE | |
| C839 | 2A16ED | LD | HL,(DEEP) | |
| C83C | EB | EX | DE,HL | |
| C83D | 211000 | LD | HL,10H | |
| C840 | CD24C1 | CALL | IDIV0 | |
| C843 | EB | EX | DE,HL | |
| C844 | D1 | POP | DE | |
| C845 | 2A4EED | LD | HL,(DIVWK) | |
| C848 | EB | EX | DE,HL | |
| C849 | 210800 | LD | HL,8 | |
| C84C | CD1EC1 | CALL | ISUB0 | |
| C84F | FA55C8 | JP | M,BACK1 | |
| C852 | C3D0C8 | JP | BACK3 | |
| ; | | | | |
| C855 | D5 | BACK1: | PUSH DE | |
| C856 | 2A18ED | LD | HL,(DEEP1) | |
| C859 | EB | EX | DE,HL | |
| C85A | 210100 | LD | HL,1 | |
| C85D | 19 | ADD | HL,DE | |
| C85E | EB | EX | DE,HL | |
| C85F | 2A32ED | LD | HL,(STACK) | |
| C862 | 19 | ADD | HL,DE | |
| C863 | D1 | POP | DE | |
| C864 | 6E | LD | L,(HL) | |
| C865 | 2600 | LD | H,0 | |
| C867 | EB | EX | DE,HL | |
| C868 | D5 | PUSH | DE | |
| C869 | 2A16ED | LD | HL,(DEEP) | |
| C86C | EB | EX | DE,HL | |
| C86D | 210100 | LD | HL,1 | (STACK+1)(STACK)の内容に (DEEP1+1)(DEEP1)の内 容を加えさらに1を加えて指定するアド レスの内容が(STACK+1)(STA CK)の内容に(DEEP+1)(DEE P)の内容を加えさらに1を加えて指定 するアドレスの内容以下の場合には(S TACK+1)(STACK)の内容に (DEEP1+1)(DEEP1)の内容 を加えさらに1を加えて指定するアドレ スにFFHを格納してメイン・ルーチン へもどる |
| C870 | 19 | ADD | HL,DE | |
| C871 | EB | EX | DE,HL | |
| C872 | 2A32ED | LD | HL,(STACK) | |
| C875 | 19 | ADD | HL,DE | |
| C876 | D1 | POP | DE | |
| C877 | 6E | LD | L,(HL) | |
| C878 | 2600 | LD | H,0 | |
| C87A | EB | EX | DE,HL | |

| | | | |
|------|--------|------|-------------|
| C87B | CD1EC1 | CALL | ISUB0 |
| C87E | FA96C8 | JP | M, BACK2 |
| C881 | 2A18ED | LD | HL, (DEEP1) |
| C884 | EB | EX | DE, HL |
| C885 | 210100 | LD | HL, 1 |
| C888 | 19 | ADD | HL, DE |
| C889 | EB | EX | DE, HL |
| C88A | 2A32ED | LD | HL, (STACK) |
| C88D | 19 | ADD | HL, DE |
| C88E | E5 | PUSH | HL |
| C88F | 21FF00 | LD | HL, 0FFH |
| C892 | EB | EX | DE, HL |
| C893 | E1 | POP | HL |
| C894 | 73 | LD | (HL), E |
| C895 | C9 | RET | |

| | | | |
|------|--------|--------|----------------|
| | | ; | |
| C896 | 2A16ED | BACK2: | LD HL, (DEEP) |
| C899 | EB | | EX DE, HL |
| C89A | 210100 | | LD HL, 1 |
| C89D | 19 | | ADD HL, DE |
| C89E | EB | | EX DE, HL |
| C89F | 2A32ED | | LD HL, (STACK) |
| C8A2 | 19 | | ADD HL, DE |
| C8A3 | E5 | | PUSH HL |
| C8A4 | D5 | | PUSH DE |
| C8A5 | 2A18ED | | LD HL, (DEEP1) |
| C8A8 | EB | | EX DE, HL |
| C8A9 | 210100 | | LD HL, 1 |
| C8AC | 19 | | ADD HL, DE |
| C8AD | EB | | EX DE, HL |
| C8AE | 2A32ED | | LD HL, (STACK) |
| C8B1 | 19 | | ADD HL, DE |
| C8B2 | D1 | | POP DE |
| C8B3 | 6E | | LD L, (HL) |
| C8B4 | 2600 | | LD H, 0 |
| C8B6 | EB | | EX DE, HL |
| C8B7 | E1 | | POP HL |
| C8B8 | 73 | | LD (HL), E |
| C8B9 | 2A18ED | | LD HL, (DEEP1) |
| C8BC | EB | | EX DE, HL |
| C8BD | 210100 | | LD HL, 1 |
| C8C0 | 19 | | ADD HL, DE |
| C8C1 | EB | | EX DE, HL |
| C8C2 | 2A32ED | | LD HL, (STACK) |
| C8C5 | 19 | | ADD HL, DE |
| C8C6 | E5 | | PUSH HL |
| C8C7 | 21FF00 | | LD HL, 0FFH |
| C8CA | EB | | EX DE, HL |
| C8CB | E1 | | POP HL |
| C8CC | 73 | | LD (HL), E |
| C8CD | C347C9 | | JP BACK5 |

(STACK+1) (STACK) の内容に (DEEP1+1) (DEEP1) の内容を加えさらに 1 を加えて指定するアドレスの内容を (STACK+1) (STACK) の内容に (DEEP+1) (DEEP) の内容を加えさらに 1 を加えて指定するアドレスに格納する

(STACK+1) (STACK) の内容に (DEEP1+1) (DEEP1) の内容を加えさらに 1 を加えて指定するアドレスに FFH を格納する

『BACK5』へジャンプする


```

;
C8D0 D5      BACK3: PUSH DE
C8D1 2A18ED  LD HL,(DEEP1)
C8D4 EB      EX DE,HL
C8D5 210100  LD HL,1
C8D8 19      ADD HL,DE
C8D9 EB      EX DE,HL
C8DA 2A32ED  LD HL,(STACK)
C8DD 19      ADD HL,DE
C8DE D1      POP DE
C8DF 6E      LD L,(HL)
C8E0 2600    LD H,0
C8E2 EB      EX DE,HL
C8E3 D5      PUSH DE
C8E4 2A16ED  LD HL,(DEEP)
C8E7 EB      EX DE,HL
C8E8 210100  LD HL,1
C8EB 19      ADD HL,DE
C8EC EB      EX DE,HL
C8ED 2A32ED  LD HL,(STACK)
C8F0 19      ADD HL,DE
C8F1 D1      POP DE
C8F2 6E      LD L,(HL)
C8F3 2600    LD H,0
C8F5 CD1EC1  CALL ISUB0
C8F8 FA10C9  JP M,BACK4
C8FB 2A18ED  LD HL,(DEEP1)
C8FE EB      EX DE,HL
C8FF 210100  LD HL,1
C902 19      ADD HL,DE
C903 EB      EX DE,HL
C904 2A32ED  LD HL,(STACK)
C907 19      ADD HL,DE
C908 E5      PUSH HL
C909 210000  LD HL,0
C90C EB      EX DE,HL
C90D E1      POP HL
C90E 73      LD (HL),E
C90F C9      RET

```

(STACK+1) (STACK)の内容に(DEEP1+1) (DEEP1)の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK)の内容に(DEEP+1) (DEEP)の内容以上の場合には(STACK+1) (STACK)の内容に(DEEP1+1) (DEEP1)の内容を加えさらに1を加えて指定するアドレスに0を格納してメイン・ルーチンへもどる

```

;
C910 2A16ED  BACK4: LD HL,(DEEP)
C913 EB      EX DE,HL
C914 210100  LD HL,1
C917 19      ADD HL,DE
C918 EB      EX DE,HL
C919 2A32ED  LD HL,(STACK)
C91C 19      ADD HL,DE
C91D E5      PUSH HL
C91E D5      PUSH DE
C91F 2A18ED  LD HL,(DEEP1)
C922 EB      EX DE,HL

```

(STACK+1) (STACK)の内容に(DEEP1+1) (DEEP1)の内容を(STACK+1) (STACK)の内容に(DEEP+1) (DEEP)の内容を加えさらに1を加えて指定するアドレスに格納する

| | | | | | |
|------|--------|--------|------|------------|--|
| C923 | 210100 | | LD | HL,1 | |
| C926 | 19 | | ADD | HL,DE | |
| C927 | EB | | EX | DE,HL | |
| C928 | 2A32ED | | LD | HL,(STACK) | |
| C92B | 19 | | ADD | HL,DE | |
| C92C | D1 | | POP | DE | |
| C92D | 6E | | LD | L,(HL) | |
| C92E | 2600 | | LD | H,0 | |
| C930 | EB | | EX | DE,HL | |
| C931 | E1 | | POP | HL | |
| C932 | 73 | | LD | (HL),E | |
| C933 | 2A18ED | | LD | HL,(DEEP1) | |
| C936 | EB | | EX | DE,HL | |
| C937 | 210100 | | LD | HL,1 | |
| C93A | 19 | | ADD | HL,DE | |
| C93B | EB | | EX | DE,HL | |
| C93C | 2A32ED | | LD | HL,(STACK) | |
| C93F | 19 | | ADD | HL,DE | |
| C940 | E5 | | PUSH | HL | |
| C941 | 210000 | | LD | HL,0 | |
| C944 | EB | | EX | DE,HL | |
| C945 | E1 | | POP | HL | |
| C946 | 73 | | LD | (HL),E | |
| C947 | 2A16ED | BACK5: | LD | HL,(DEEP) | |
| C94A | EB | | EX | DE,HL | |
| C94B | 210000 | | LD | HL,0 | |
| C94E | CD1EC1 | | CALL | ISUB0 | |
| C951 | B3 | | OR | E | |
| C952 | C276C9 | | JP | NZ,BACK6 | |
| C955 | D5 | | PUSH | DE | |
| C956 | 210000 | | LD | HL,0 | |
| C959 | EB | | EX | DE,HL | |
| C95A | 2A32ED | | LD | HL,(STACK) | |
| C95D | 19 | | ADD | HL,DE | |
| C95E | D1 | | POP | DE | |
| C95F | 6E | | LD | L,(HL) | |
| C960 | 2600 | | LD | H,0 | |
| C962 | 221CED | | LD | (WORK4),HL | |
| C965 | D5 | | PUSH | DE | |
| C966 | 2A1CED | | LD | HL,(WORK4) | |
| C969 | EB | | EX | DE,HL | |
| C96A | 2A32ED | | LD | HL,(STACK) | |
| C96D | 19 | | ADD | HL,DE | |
| C96E | D1 | | POP | DE | |
| C96F | 6E | | LD | L,(HL) | |
| C970 | 2600 | | LD | H,0 | |
| C972 | 2234ED | | LD | (COORD),HL | |
| C975 | C9 | | RET | | |
| C976 | 2A18ED | BACK6: | LD | HL,(DEEP1) | |
| C979 | EB | | EX | DE,HL | |

(STACK+1)(STACK)の内容に(DEEP1+1)(DEEP1)の内容を加えさらに1を加えて指定するアドレスに0を格納する

(DEEP+1)(DEEP)の内容が0の場合には(STACK+1)(STACK)で指定するアドレスの内容を(WORK4+1)(WORK4)に格納しさらにその値に(STACK+1)(STACK)の内容を加えて指定するアドレスの内容を(COORD+1)(COORD)に格納してメイン・ルーチンへもどる

| | | | | |
|------|--------|--------|------------|--|
| C97A | 211000 | LD | HL,10H | { (DEEP1+1) (DEEP1) の内容から16を減じる |
| C97D | CD1EC1 | CALL | ISUB0 | |
| C980 | EB | EX | DE,HL | { (DEEP1+1) (DEEP1) の内容が0未満の場合にはメイン・ルーチンへもどる |
| C981 | 2218ED | LD | (DEEP1),HL | |
| C984 | 2A18ED | LD | HL,(DEEP1) | { (DEEP1+1) (DEEP1) の内容が0未満の場合にはメイン・ルーチンへもどる |
| C987 | EB | EX | DE,HL | |
| C988 | 210000 | LD | HL,0 | { (DEEP1+1) (DEEP1) の内容が0未満の場合にはメイン・ルーチンへもどる |
| C98B | CD1EC1 | CALL | ISUB0 | |
| C98E | F292C9 | JP | P,BACK7 | { (DEEP1+1) (DEEP1) の内容が0未満の場合にはメイン・ルーチンへもどる |
| C991 | C9 | RET | | |
| ; | | | | |
| C992 | D5 | BACK7: | PUSH DE | { (DEEP+1) (DEEP) の内容を16で割った剰余が8以上の場合には『BACK10』へジャンプする |
| C993 | 2A16ED | LD | HL,(DEEP) | |
| C996 | EB | EX | DE,HL | { (DEEP+1) (DEEP) の内容を16で割った剰余が8以上の場合には『BACK10』へジャンプする |
| C997 | 211000 | LD | HL,10H | |
| C99A | CD24C1 | CALL | IDIV0 | { (DEEP+1) (DEEP) の内容を16で割った剰余が8以上の場合には『BACK10』へジャンプする |
| C99D | EB | EX | DE,HL | |
| C99E | D1 | POP | DE | { (DEEP+1) (DEEP) の内容を16で割った剰余が8以上の場合には『BACK10』へジャンプする |
| C99F | 2A4EED | LD | HL,(DIVWK) | |
| C9A2 | EB | EX | DE,HL | { (DEEP+1) (DEEP) の内容を16で割った剰余が8以上の場合には『BACK10』へジャンプする |
| C9A3 | 210800 | LD | HL,8 | |
| C9A6 | CD1EC1 | CALL | ISUB0 | { (DEEP+1) (DEEP) の内容を16で割った剰余が8以上の場合には『BACK10』へジャンプする |
| C9A9 | FAAFC9 | JP | M,BACK8 | |
| C9AC | C3E4C9 | JP | BACK10 | |
| ; | | | | |
| C9AF | D5 | BACK8: | PUSH DE | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9B0 | 2A16ED | LD | HL,(DEEP) | |
| C9B3 | EB | EX | DE,HL | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9B4 | 210100 | LD | HL,1 | |
| C9B7 | 19 | ADD | HL,DE | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9B8 | EB | EX | DE,HL | |
| C9B9 | 2A32ED | LD | HL,(STACK) | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9BC | 19 | ADD | HL,DE | |
| C9BD | D1 | POP | DE | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9BE | 6E | LD | L,(HL) | |
| C9BF | 2600 | LD | H,0 | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9C1 | EB | EX | DE,HL | |
| C9C2 | D5 | PUSH | DE | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9C3 | 2A18ED | LD | HL,(DEEP1) | |
| C9C6 | EB | EX | DE,HL | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9C7 | 210100 | LD | HL,1 | |
| C9CA | 19 | ADD | HL,DE | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9CB | EB | EX | DE,HL | |
| C9CC | 2A32ED | LD | HL,(STACK) | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9CF | 19 | ADD | HL,DE | |
| C9D0 | D1 | POP | DE | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9D1 | 6E | LD | L,(HL) | |
| C9D2 | 2600 | LD | H,0 | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9D4 | CD1EC1 | CALL | ISUB0 | |
| C9D7 | F2DDC9 | JP | P,BACK9 | { (STACK+1) (STACK) の内容に(DEEP+1) (DEEP) の内容を加えさらに1を加えて指定するアドレスの内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定するアドレスの内容未満の場合には『BACK6』へもどってくり返す |
| C9DA | C376C9 | JP | BACK6 | |

| | | | | | |
|-------------|--------|---------|------|-------------|--|
| C9DD | 210100 | BACK9: | LD | HL,1 | } (ABFLAG+1) (ABFLAG) を 1にセットしてメイン・ルーチンへもど る |
| C9E0 | 222CED | | LD | (ABFLAG),HL | |
| C9E3 | C9 | | RET | | |
| ; | | | | | |
| C9E4 | D5 | BACK10: | PUSH | DE | } (STACK+1) (STACK) の内容 に(DEEP+1) (DEEP) の内容を 加えさらに1を加えて指定するアドレス の内容が(STACK+1) (STACK) の内容に(DEEP1+1) (DEEP1) の内容を加えさらに1を加えて指定する アドレスの内容を越える場合には『BA CK』へもどってくり返す |
| C9E5 | 2A16ED | | LD | HL,(DEEP) | |
| C9E8 | EB | | EX | DE,HL | |
| C9E9 | 210100 | | LD | HL,1 | |
| C9EC | 19 | | ADD | HL,DE | |
| C9ED | EB | | EX | DE,HL | |
| C9EE | 2A32ED | | LD | HL,(STACK) | |
| C9F1 | 19 | | ADD | HL,DE | |
| C9F2 | D1 | | POP | DE | |
| C9F3 | 6E | | LD | L,(HL) | |
| C9F4 | 2600 | | LD | H,0 | |
| C9F6 | EB | | EX | DE,HL | |
| C9F7 | D5 | | PUSH | DE | |
| C9F8 | 2A18ED | | LD | HL,(DEEP1) | |
| C9FB | EB | | EX | DE,HL | |
| C9FC | 210100 | | LD | HL,1 | |
| C9FF | 19 | | ADD | HL,DE | |
| CA00 | EB | | EX | DE,HL | |
| CA01 | 2A32ED | | LD | HL,(STACK) | |
| CA04 | 19 | | ADD | HL,DE | |
| CA05 | D1 | | POP | DE | |
| CA06 | 6E | | LD | L,(HL) | |
| CA07 | 2600 | | LD | H,0 | |
| CA09 | EB | | EX | DE,HL | |
| CA0A | CD1EC1 | | CALL | ISUB0 | |
| CA0D | F213CA | | JP | P,BACK11 | |
| CA10 | C376C9 | | JP | BACK6 | |
| ; | | | | | |
| CA13 | C3DDC9 | BACK11: | JP | BACK9 | } 『BACK9』へもどってくり返す |
| ; | | | | | |
| ; animation | | | | | |
| ; | | | | | |
| CA16 | 210000 | DISP: | LD | HL,0 | } (COORD2+1) (COORD2) に 0を格納する |
| CA19 | 2212ED | | LD | (COORD2),HL | |
| CA1C | 2A10ED | | LD | HL,(COORD1) | } (COORD1+1) (COORD1) の内 容を4倍して2を加えたものを (DEE P+1) (DEEP) に格納する |
| CA1F | EB | | EX | DE,HL | |
| CA20 | 210400 | | LD | HL,4 | |
| CA23 | CD21C1 | | CALL | IMULT0 | |
| CA26 | 210200 | | LD | HL,2 | |
| CA29 | 19 | | ADD | HL,DE | |
| CA2A | 2216ED | | LD | (DEEP),HL | |
| CA2D | D5 | | PUSH | DE | |
| CA2E | 210600 | | LD | HL,6 | |
| CA31 | EB | | EX | DE,HL | |
| CA32 | 2A10ED | | LD | HL,(COORD1) | |
| CA35 | 19 | | ADD | HL,DE | |

| | | | | |
|------|--------|--------|----------------|---|
| CA36 | EB | EX | DE,HL | } (BOARD+1) (BOARD) の内容 に6を加えさらに(COORD1+1) (COORD1)の内容を加えて指定する アドレスの内容が0を越える場合には 『DISP9』へジャンプする |
| CA37 | 2A38ED | LD | HL,(BOARD) | |
| CA3A | 19 | ADD | HL,DE | |
| CA3B | D1 | POP | DE | |
| CA3C | 6E | LD | L,(HL) | |
| CA3D | 2600 | LD | H,0 | |
| CA3F | EB | EX | DE,HL | |
| CA40 | 210000 | LD | HL,0 | |
| CA43 | EB | EX | DE,HL | |
| CA44 | CD1EC1 | CALL | ISUB0 | |
| CA47 | F24DCA | JP | P,DISP1 | |
| CA4A | C317CB | JP | DISP9 | |
| ; | | | | |
| CA4D | 110000 | DISP1: | LD DE,0 | } プログラム・カウンタ(PC)の値をスタ ックへ退避する |
| CA50 | CD53CA | | CALL DISP2 | |
| CA53 | D5 | DISP2: | PUSH DE | } 0をスタックへ退避する |
| CA54 | 2A12ED | | LD HL,(COORD2) | |
| CA57 | EB | | EX DE,HL | } (DEEP1+1) (DEEP1) をルー プ・カウンタとして(COORD2+1) (COORD2)の内容を3倍して6を加 えた値から3倍して8を加えた値まで3 回のくり返し処理を行う |
| CA58 | 210300 | | LD HL,3 | |
| CA5B | CD21C1 | | CALL IMULT0 | |
| CA5E | 210600 | | LD HL,6 | |
| CA61 | 19 | | ADD HL,DE | |
| CA62 | 2218ED | | LD (DEEP1),HL | |
| CA65 | 2A12ED | | LD HL,(COORD2) | |
| CA68 | EB | | EX DE,HL | |
| CA69 | 210300 | | LD HL,3 | |
| CA6C | CD21C1 | | CALL IMULT0 | |
| CA6F | 210800 | | LD HL,8 | } 1個の石を表示する |
| CA72 | 19 | | ADD HL,DE | |
| CA73 | EB | | EX DE,HL | |
| CA74 | CD77CA | | CALL DISP3 | |
| CA77 | D5 | DISP3: | PUSH DE | |
| CA78 | CD46CB | | CALL DSUB | |
| CA7B | 210100 | | LD HL,1 | |
| CA7E | 221CED | | LD (WORK4),HL | |
| CA81 | 21F401 | | LD HL,500 | |
| CA84 | EB | | EX DE,HL | |
| CA85 | CD88CA | | CALL DISP4 | } (WORK4+1) (WORK4) をルー プ・カウンタとして500回のくり返し 処理を行い時間を浪費する |
| CA88 | D5 | DISP4: | PUSH DE | |
| CA89 | 2A1CED | | LD HL,(WORK4) | |
| CA8C | EB | | EX DE,HL | |
| CA8D | 210100 | | LD HL,1 | |
| CA90 | 19 | | ADD HL,DE | |
| CA91 | 221CED | | LD (WORK4),HL | |
| CA94 | D1 | | POP DE | |
| CA95 | D5 | | PUSH DE | |
| CA96 | CD1EC1 | | CALL ISUB0 | |
| CA99 | D1 | | POP DE | |
| CA9A | E1 | | POP HL | |
| CA9B | FAA0CA | | JP M,DISP5 | |
| CA9E | E5 | | PUSH HL | |

| | | | | | |
|------|--------|--------|--------|----------------|---|
| CA9F | E9 | | JP | (HL) | |
| CAA0 | CD37CC | ; | DISP5: | CALL DERA1 | } 石の上段を消去する |
| CAA3 | 2A18ED | | | LD HL,(DEEP1) | |
| CAA6 | EB | | | EX DE,HL | } (DEEP1+1)(DEEP1)をループ・カウンタとするくり返し処理の終了をチェックする |
| CAA7 | 210100 | | | LD HL,1 | |
| CAAA | 19 | | | ADD HL,DE | |
| CAAB | 2218ED | | | LD (DEEP1),HL | |
| CAAE | D1 | | | POP DE | |
| CAAF | D5 | | | PUSH DE | |
| CAB0 | CD1EC1 | | | CALL ISUB0 | |
| CAB3 | D1 | | | POP DE | |
| CAB4 | E1 | | | POP HL | |
| CAB5 | FABACA | | | JP M,DISP6 | |
| CAB8 | E5 | | | PUSH HL | |
| CAB9 | E9 | | | JP (HL) | |
| CABA | 2A12ED | ; | DISP6: | LD HL,(COORD2) | } (COORD2+1)(COORD2)の内容に1を加える |
| CABD | EB | | | EX DE,HL | |
| CABE | 210100 | | | LD HL,1 | } (COORD2+1)(COORD2)の内容に1を加える |
| CAC1 | 19 | | | ADD HL,DE | |
| CAC2 | 2212ED | | | LD (COORD2),HL | |
| CAC5 | D5 | | | PUSH DE | |
| CAC6 | D5 | | | PUSH DE | |
| CAC7 | 2A12ED | | | LD HL,(COORD2) | |
| CACA | EB | | | EX DE,HL | |
| CACB | 210100 | | | LD HL,1 | |
| CACE | 19 | | | ADD HL,DE | |
| CACF | EB | | | EX DE,HL | |
| CAD0 | 210600 | | | LD HL,6 | } (BOARD+1)(BOARD)の内容に(COORD2+1)(COORD2)の内容を加え1を加え6倍してさらに(COORD1+1)(COORD1)の内容を加えて指定するアドレスの内容が0以下でかつ(COORD2+1)(COORD2)の内容が4以下の場合にのみ『DISP2』にもどってくり返す |
| CAD3 | CD21C1 | | | CALL IMULT0 | |
| CAD6 | 2A10ED | | | LD HL,(COORD1) | |
| CAD9 | 19 | | | ADD HL,DE | |
| CADA | EB | | | EX DE,HL | |
| CADB | 2A38ED | | | LD HL,(BOARD) | |
| CADE | 19 | | | ADD HL,DE | |
| CADF | D1 | | | POP DE | |
| CAE0 | 6E | | | LD L,(HL) | |
| CAE1 | 2600 | | | LD H,0 | |
| CAE3 | EB | | | EX DE,HL | } (BOARD+1)(BOARD)の内容に(COORD2+1)(COORD2)の内容を加え1を加え6倍してさらに(COORD1+1)(COORD1)の内容を加えて指定するアドレスの内容が0以下でかつ(COORD2+1)(COORD2)の内容が4以下の場合にのみ『DISP2』にもどってくり返す |
| CAE4 | 210000 | | | LD HL,0 | |
| CAE7 | EB | | | EX DE,HL | |
| CAE8 | CD1EC1 | | | CALL ISUB0 | |
| CAEB | 110000 | | | LD DE,0 | |
| CAEE | F2F2CA | | | JP P,DISP7 | |
| CAF1 | 1C | | | INC E | |
| CAF2 | EB | DISP7: | | EX DE,HL | |
| CAF3 | D1 | | | POP DE | |
| CAF4 | EB | | | EX DE,HL | |
| CAF5 | D5 | | | PUSH DE | |
| CAF6 | 2A12ED | | | LD HL,(COORD2) | |

| | | | |
|-------------|-------------------|----------------|---------------------|
| CAF9 EB | | EX DE,HL | |
| CAFA 210400 | | LD HL,4 | |
| CAFD EB | | EX DE,HL | |
| CAFE CD1EC1 | | CALL ISUB0 | |
| CB01 110000 | | LD DE,0 | |
| CB04 F208CB | | JP P,DISP8 | |
| CB07 1C | | INC E | |
| CB08 EB | DISP8: | EX DE,HL | |
| CB09 D1 | | POP DE | |
| CB0A 19 | | ADD HL,DE | |
| CB0B D1 | | POP DE | |
| CB0C D5 | | PUSH DE | |
| CB0D CD1EC1 | | CALL ISUB0 | |
| CB10 D1 | | POP DE | |
| CB11 E1 | | POP HL | |
| CB12 FA17CB | | JP M,DISP9 | |
| CB15 E5 | | PUSH HL | |
| CB16 E9 | | JP (HL) | |
| | | | |
| CB17 2A12ED | ; DISP9: | LD HL,(COORD2) | |
| CB1A EB | | EX DE,HL | |
| CB1B 210300 | | LD HL,3 | (COORD2+1)(COORD2)の |
| CB1E CD21C1 | | CALL IMULT0 | 内容を3倍し6を加えた値を(DEEP |
| CB21 210600 | | LD HL,6 | 1+1)(DEEP1)に格納する |
| CB24 19 | | ADD HL,DE | |
| CB25 2218ED | | LD (DEEP1),HL | |
| CB28 CD46CB | | CALL DSUB | 最終位置に石を表示する |
| CB2B 2A12ED | | LD HL,(COORD2) | |
| CB2E EB | | EX DE,HL | |
| CB2F 210600 | | LD HL,6 | |
| CB32 CD21C1 | | CALL IMULT0 | |
| CB35 2A10ED | | LD HL,(COORD1) | (BOARD+1)(BOARD)の内容 |
| CB38 19 | | ADD HL,DE | に(COORD2+1)(COORD2) |
| CB39 EB | | EX DE,HL | の内容を加え6倍してさらに(COOR |
| CB3A 2A38ED | | LD HL,(BOARD) | D1+1)(COORD1)の内容を加え |
| CB3D 19 | | ADD HL,DE | て指定するアドレスに(STONE+1) |
| CB3E E5 | | PUSH HL | (STONE)の内容を格納する |
| CB3F 2A06ED | | LD HL,(STONE) | |
| CB42 EB | | EX DE,HL | |
| CB43 E1 | | POP HL | |
| CB44 73 | | LD (HL),E | |
| CB45 C9 | | RET | メイン・ルーチンへもどる |
| | | | |
| | ; display a stone | | |
| | ; DSUB: | LD HL,(STONE) | |
| CB46 2A06ED | | EX DE,HL | |
| CB49 EB | | LD HL,1 | |
| CB4A 210100 | | CALL ISUB0 | |
| CB4D CD1EC1 | | OR E | |
| CB50 B3 | | JP NZ,DSUB1 | |
| CB51 C263CB | | | |

| | | | | | |
|-------|----------|--------|------|------------|---|
| CB54 | 210000 | | LD | HL,0 | (STONE+1)(STONE)の内容 が1の場合にはカラーを黄色に設定する |
| CB57 | EB | | EX | DE,HL | |
| CB58 | 2A2EED | | LD | HL,(PATR) | |
| CB5B | 19 | | ADD | HL,DE | |
| CB5C | E5 | | PUSH | HL | |
| CB5D | 21C800 | | LD | HL,0C8H | |
| CB60 | EB | | EX | DE,HL | |
| CB61 | E1 | | POP | HL | |
| CB62 | 73 | | LD | (HL),E | |
| CB63 | 2A06ED | DSUB1: | LD | HL,(STONE) | |
| CB66 | EB | | EX | DE,HL | (STONE+1)(STONE)の内容 が2の場合にはカラーを水色に設定する |
| CB67 | 210200 | | LD | HL,2 | |
| CB6A | CD1EC1 | | CALL | ISUB0 | |
| CB6D | B3 | | OR | E | |
| CB6E | C280CB | | JP | NZ,DSUB2 | |
| CB71 | 210000 | | LD | HL,0 | |
| CB74 | EB | | EX | DE,HL | |
| CB75 | 2A2EED | | LD | HL,(PATR) | |
| CB78 | 19 | | ADD | HL,DE | |
| CB79 | E5 | | PUSH | HL | |
| CB7A | 21A800 | | LD | HL,0A8H | カーソルの横位置を(DEEP+1)(DEEP)の内容によって指定する |
| CB7D | EB | | EX | DE,HL | |
| CB7E | E1 | | POP | HL | |
| CB7F | 73 | | LD | (HL),E | |
| CB80 | 210100 | DSUB2: | LD | HL,1 | |
| CB83 | EB | | EX | DE,HL | |
| CB84 | 2A36ED | | LD | HL,(PCUS) | |
| CB87 | 19 | | ADD | HL,DE | |
| CB88 | E5 | | PUSH | HL | |
| CB89 | 2A16ED | | LD | HL,(DEEP) | |
| CB8C | EB | | EX | DE,HL | カーソルの縦位置を(DEEP1+1)(DEEP1)の内容によって指定する |
| CB8D | E1 | | POP | HL | |
| CB8E | 73 | | LD | (HL),E | |
| CB8F | 210000 | | LD | HL,0 | |
| CB92 | EB | | EX | DE,HL | |
| CB93 | 2A36ED | | LD | HL,(PCUS) | |
| CB96 | 19 | | ADD | HL,DE | |
| CB97 | E5 | | PUSH | HL | |
| CB98 | 2A18ED | | LD | HL,(DEEP1) | |
| CB9B | EB | | EX | DE,HL | |
| CB9C | E1 | | POP | HL | 石の上段を表示する |
| CB9D | 73 | | LD | (HL),E | |
| CB9E | C3A5CB | | JP | DSUB3 | |
| CBA1 | 86868686 | MSG1: | DB | '■■■■' | |
| CBA5 | 0603 | DSUB3: | LD | B,3 | |
| CBA7 | 21A1CB | | LD | HL,MSG1 | |
| CBA A | 7E | DSUB4: | LD | A,(HL) | |
| CBAB | CD39C1 | | CALL | OTCHRO | |
| CBAE | 23 | | INC | HL | |

| | | | |
|-------------|-----------|---------------|--|
| CBAF 10F9 | | DJNZ DSUB4 | |
| | ; | | |
| CBB1 210100 | | LD HL,1 | カーソルの横位置を(DEEP+1)(DEEP)の内容によって指定する |
| CBB4 EB | | EX DE,HL | |
| CBB5 2A36ED | | LD HL,(PCUS) | |
| CBB8 19 | | ADD HL,DE | |
| CBB9 E5 | | PUSH HL | |
| CBBA 2A16ED | | LD HL,(DEEP) | カーソルの縦位置を(DEEP1+1)(DEEP1)の内容に1を加えた値に指定する |
| CBBD EB | | EX DE,HL | |
| CBBE E1 | | POP HL | |
| CBBF 73 | | LD (HL),E | |
| CBC0 210000 | | LD HL,0 | |
| CBC3 EB | | EX DE,HL | 石の中段を表示する |
| CBC4 2A36ED | | LD HL,(PCUS) | |
| CBC7 19 | | ADD HL,DE | |
| CBC8 E5 | | PUSH HL | |
| CBC9 2A18ED | | LD HL,(DEEP1) | |
| CBCC EB | | EX DE,HL | カーソルの横位置を(DEEP+1)(DEEP)の内容によって指定する |
| CBCD 210100 | | LD HL,1 | |
| CBD0 19 | | ADD HL,DE | |
| CBD1 EB | | EX DE,HL | |
| CBD2 E1 | | POP HL | |
| CBD3 73 | | LD (HL),E | カーソルの縦位置を(DEEP1+1)(DEEP1)の内容に2を加えた値に指定する |
| CBD4 C3DACB | | JP DSUB5 | |
| | ; | | |
| CBD7 878787 | MSG2: DB | '■■■■' | |
| | ; | | |
| CBDA 0603 | DSUB5: LD | B,3 | カーソルの横位置を(DEEP+1)(DEEP)の内容によって指定する |
| CBDC 21D7CB | | LD HL,MSG2 | |
| CBDF 7E | DSUB6: LD | A,(HL) | |
| CBE0 CD39C1 | | CALL OTCHR0 | |
| CBE3 23 | | INC HL | |
| CBE4 10F9 | | DJNZ DSUB6 | カーソルの縦位置を(DEEP1+1)(DEEP1)の内容に2を加えた値に指定する |
| | ; | | |
| CBE6 210100 | | LD HL,1 | |
| CBE9 EB | | EX DE,HL | |
| CBEA 2A36ED | | LD HL,(PCUS) | |
| CBED 19 | | ADD HL,DE | カーソルの横位置を(DEEP+1)(DEEP)の内容によって指定する |
| CBEE E5 | | PUSH HL | |
| CBEF 2A16ED | | LD HL,(DEEP) | |
| CBF2 EB | | EX DE,HL | |
| CBF3 E1 | | POP HL | |
| CBF4 73 | | LD (HL),E | カーソルの縦位置を(DEEP1+1)(DEEP1)の内容に2を加えた値に指定する |
| CBF5 210000 | | LD HL,0 | |
| CBF8 EB | | EX DE,HL | |
| CBF9 2A36ED | | LD HL,(PCUS) | |
| CBFC 19 | | ADD HL,DE | |
| CBFD E5 | | PUSH HL | カーソルの横位置を(DEEP+1)(DEEP)の内容によって指定する |
| CBFE 2A18ED | | LD HL,(DEEP1) | |
| CC01 EB | | EX DE,HL | |
| CC02 210200 | | LD HL,2 | |

| | | | | |
|------|--------|--------|--------------------|------------|
| CC05 | 19 | ADD | HL,DE | |
| CC06 | EB | EX | DE,HL | |
| CC07 | E1 | POP | HL | |
| CC08 | 73 | LD | (HL),E | |
| CC09 | C30FCC | JP | DSUB7 | |
| | | ; | | |
| CC0C | 878787 | MSG3: | DB | '■■■■' |
| | | ; | | |
| CC0F | 0603 | DSUB7: | LD | B,3 |
| CC11 | 210CCC | | LD | HL,MSG3 |
| CC14 | 7E | DSUB8: | LD | A,(HL) |
| CC15 | CD39C1 | | CALL | OTCHRO |
| CC18 | 23 | | INC | HL |
| CC19 | 10F9 | | DJNZ | DSUB8 |
| | | ; | | |
| CC1B | 210000 | | LD | HL,0 |
| CC1E | EB | | EX | DE,HL |
| CC1F | 2A2EED | | LD | HL,(PATR) |
| CC22 | 19 | | ADD | HL,DE |
| CC23 | E5 | | PUSH | HL |
| CC24 | 21E800 | | LD | HL,0E8H |
| CC27 | EB | | EX | DE,HL |
| CC28 | E1 | | POP | HL |
| CC29 | 73 | | LD | (HL),E |
| CC2A | C9 | | RET | |
| | | ; | | |
| | | ; | erase 3 characters | |
| | | ; | | |
| CC2B | 212400 | DERA: | LD | HL,24H |
| CC2E | 2216ED | | LD | (DEEP),HL |
| CC31 | 211400 | | LD | HL,14H |
| CC34 | 2218ED | | LD | (DEEP1),HL |
| CC37 | 210100 | DERA1: | LD | HL,1 |
| CC3A | EB | | EX | DE,HL |
| CC3B | 2A36ED | | LD | HL,(PCUS) |
| CC3E | 19 | | ADD | HL,DE |
| CC3F | E5 | | PUSH | HL |
| CC40 | 2A16ED | | LD | HL,(DEEP) |
| CC43 | EB | | EX | DE,HL |
| CC44 | E1 | | POP | HL |
| CC45 | 73 | | LD | (HL),E |
| CC46 | 210000 | | LD | HL,0 |
| CC49 | EB | | EX | DE,HL |
| CC4A | 2A36ED | | LD | HL,(PCUS) |
| CC4D | 19 | | ADD | HL,DE |
| CC4E | E5 | | PUSH | HL |
| CC4F | 2A18ED | | LD | HL,(DEEP1) |
| CC52 | EB | | EX | DE,HL |
| CC53 | E1 | | POP | HL |
| CC54 | 73 | | LD | (HL),E |
| CC55 | C35BCC | | JP | DERA2 |

石の下段を表示する

カラーを白色にもどす

メインルーチンへもどる

カーソル位置を座標(36, 20)に指定する

上記指定位置までカーソルを移動する


```

;
CC58 202020 MSG4: DB ' ' ' '
;
CC5B 0603 DERA2: LD B,3
CC5D 2158CC LD HL,MSG4
CC60 7E DERA3: LD A,(HL)
CC61 CD39C1 CALL OTCHRO
CC64 23 INC HL
CC65 10F9 DJNZ DERA3
;
CC67 C9 RET
;
; ring the bell
;
CC68 3E20 BELL: LD A,20H
CC6A D340 OUT (40H),A
CC6C 210100 LD HL,1
CC6F 221CED LD (WORK4),HL
CC72 21F401 LD HL,500
CC75 EB EX DE,HL
CC76 CD79CC CALL BELL1
CC79 D5 BELL1: PUSH DE
CC7A 2A1CED LD HL,(WORK4)
CC7D EB EX DE,HL
CC7E 210100 LD HL,1
CC81 19 ADD HL,DE
CC82 221CED LD (WORK4),HL
CC85 D1 POP DE
CC86 D5 PUSH DE
CC87 CD1EC1 CALL ISUB0
CC8A D1 POP DE
CC8B E1 POP HL
CC8C FA91CC JP M,BELL2
CC8F E5 PUSH HL
CC90 E9 JP (HL)
;
CC91 AF BELL2: XOR A
CC92 D340 OUT (40H),A
CC94 C9 RET
;
CC95 END

```

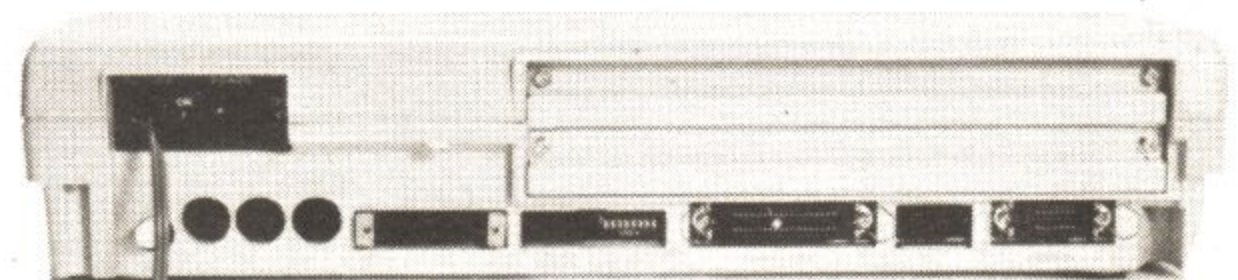
3個の空白を表示する

メインルーチンへもどる

ブザーの鳴動を開始する

(WORK4+1)(WORK4)をループ・カウンタとして500回のくり返し処理を行い時間を浪費する

ブザーの鳴動を停止する



▲PC-8001mkII本体背面

◀PC-8001mkII本体正面

メインルーチン・アセンブル・リスト

《第118図》2次元4目ならべ・メインルーチン・アセンブル・リスト

```

;
;*****
;
; 2d4m main routine
;
;*****
;
08F7 SETCON:EQU 08F7H
093A SETWID:EQU 093AH
5C66 MONHOT:EQU 5C66H
C103 CHAR0: EQU 0C103H
C106 TAB0: EQU 0C106H
C109 CRLF0: EQU 0C109H
C112 INKEY0:EQU 0C112H
C11E ISUB0: EQU 0C11EH
C121 IMULT0:EQU 0C121H
C124 IDIVO: EQU 0C124H
C12A MINUS0:EQU 0C12AH
C12D RND0: EQU 0C12DH
C139 OTCHRO:EQU 0C139H
C400 EMPTY: EQU 0C400H
C503 LIST: EQU 0C503H
C5CE VAL: EQU 0C5CEH
C810 BACK: EQU 0C810H
CA16 DISP: EQU 0CA16H
CB46 DSUB: EQU 0CB46H
CC2A DERA: EQU 0CC2AH
CC67 BELL: EQU 0CC67H
EA5B ATRCOD:EQU 0EA5BH
EA5D ROLLIN:EQU 0EA5DH
EA63 CUSPOS:EQU 0EA63H
ED06 SOTNE: EQU 0ED06H
ED08 PLAYS: EQU 0ED08H

```


| | | | |
|-------------|----------------|--------------|--|
| ED0A | PLAYER:EQU | 0ED0AH | |
| ED0C | STONE1:EQU | 0ED0CH | |
| ED10 | COORD1:EQU | 0ED10H | |
| ED12 | COORD2:EQU | 0ED12H | |
| ED16 | DEEP: EQU | 0ED16H | |
| ED18 | DEEP1: EQU | 0ED18H | |
| ED1C | WORK4: EQU | 0ED1CH | |
| ED28 | VALUE: EQU | 0ED28H | |
| ED2A | RDEEP: EQU | 0ED2AH | |
| ED2C | ABFLAG:EQU | 0ED2CH | |
| ED2E | PATR: EQU | 0ED2EH | |
| ED30 | VECTOR:EQU | 0ED30H | |
| ED32 | STACK: EQU | 0ED32H | |
| ED34 | COORD: EQU | 0ED34H | |
| ED36 | PCUS: EQU | 0ED36H | |
| ED38 | BOARD: EQU | 0ED38H | |
| ED4E | DIVWK: EQU | 0ED4EH | |
| | ; | | |
| | ORG | 0CC94H | |
| | ; | | |
| | ; main routine | | |
| | ; | | |
| CC94 011928 | MAIN: LD | BC,2819H | } 36桁×25行モードに設定する |
| CC97 CD3A09 | CALL | SETWID | |
| CC9A 211801 | LD | HL,0118H | } スクリーン全体をスクロール・エリアに設定する |
| CC9D 225DEA | LD | (ROLL IN),HL | |
| CCA0 01FF00 | LD | BC,0FFH | } カラー・モードでファンクション・キー表示無に設定する |
| CCA3 CDF708 | CALL | SETCON | |
| | ; | | |
| CCA6 2100C4 | LD | HL,EMPTY | } 評価テーブルの先頭アドレスを (BOARD+1) (BOARD) に格納する |
| CCA9 2238ED | LD | (BOARD),HL | |
| CCAC 2A38ED | LD | HL,(BOARD) | } (BOARD+1) (BOARD) に36を加えたものを (VECTOR+1) (VECTOR) に格納する |
| CCAF EB | EX | DE,HL | |
| CCB0 212400 | LD | HL,24H | } (VECTOR+1) (VECTOR) に8を加えたものを (STACK+1) (STACK) に格納する |
| CCB3 19 | ADD | HL,DE | |
| CCB4 2230ED | LD | (VECTOR),HL | } カーソル指定に使用するワーク・エリアのアドレスを (PCUS+1) (PCUS) に格納する |
| CCB7 2A30ED | LD | HL,(VECTOR) | |
| CCBA EB | EX | DE,HL | } カラー指定に使用するワーク・エリアのアドレスを (PATR+1) (PATR) に格納する |
| CCBB 210800 | LD | HL,8 | |
| CCBE 19 | ADD | HL,DE | } スクリーンの最大桁を80桁に設定する |
| CCBF 2232ED | LD | (STACK),HL | |
| CCC2 2163EA | LD | HL,CUSPOS | |
| CCC5 2236ED | LD | (PCUS),HL | |
| CCC8 215BEA | LD | HL,ATRCOD | |
| CCCB 222EED | LD | (PATR),HL | |
| CCCE 210100 | LD | HL,1 | |
| CCD1 EB | EX | DE,HL | |
| CCD2 2A2EED | LD | HL,(PATR) | |
| CCD5 19 | ADD | HL,DE | |
| CCD6 E5 | PUSH | HL | |
| CCD7 215000 | LD | HL,50H | |

| | | | |
|-------------|----------------|---|---|
| CCDA EB | EX DE,HL | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに1を格納する |
| CCDB E1 | POP HL | | |
| CCDC 73 | LD (HL),E | | |
| CCDD 210000 | LD HL,0 | | |
| CCE0 EB | EX DE,HL | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに1を格納する |
| CCE1 2A30ED | LD HL,(VECTOR) | | |
| CCE4 19 | ADD HL,DE | | |
| CCE5 E5 | PUSH HL | | |
| CCE6 210100 | LD HL,1 | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに1を加えたアドレスに0を格納する |
| CCE9 EB | EX DE,HL | | |
| CCEA E1 | POP HL | | |
| CCEB 73 | LD (HL),E | | |
| CCEC 210100 | LD HL,1 | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに1を加えたアドレスに0を格納する |
| CCEF EB | EX DE,HL | | |
| CCF0 2A30ED | LD HL,(VECTOR) | | |
| CCF3 19 | ADD HL,DE | | |
| CCF4 E5 | PUSH HL | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに2を加えたアドレスに1を格納する |
| CCF5 210000 | LD HL,0 | | |
| CCF8 EB | EX DE,HL | | |
| CCF9 E1 | POP HL | | |
| CCFA 73 | LD (HL),E | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに2を加えたアドレスに1を格納する |
| CCFB 210200 | LD HL,2 | | |
| CCFE EB | EX DE,HL | | |
| CCFF 2A30ED | LD HL,(VECTOR) | | |
| CD02 19 | ADD HL,DE | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに3を加えたアドレスに1を格納する |
| CD03 E5 | PUSH HL | | |
| CD04 210100 | LD HL,1 | | |
| CD07 EB | EX DE,HL | | |
| CD08 E1 | POP HL | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに3を加えたアドレスに1を格納する |
| CD09 73 | LD (HL),E | | |
| CD0A 210300 | LD HL,3 | | |
| CD0D EB | EX DE,HL | | |
| CD0E 2A30ED | LD HL,(VECTOR) | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに4を加えたアドレスに0を格納する |
| CD11 19 | ADD HL,DE | | |
| CD12 E5 | PUSH HL | | |
| CD13 210100 | LD HL,1 | | |
| CD16 EB | EX DE,HL | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに4を加えたアドレスに0を格納する |
| CD17 E1 | POP HL | | |
| CD18 73 | LD (HL),E | | |
| CD19 210400 | LD HL,4 | | |
| CD1C EB | EX DE,HL | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに4を加えたアドレスに0を格納する |
| CD1D 2A30ED | LD HL,(VECTOR) | | |
| CD20 19 | ADD HL,DE | | |
| CD21 E5 | PUSH HL | | |
| CD22 210000 | LD HL,0 | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに4を加えたアドレスに0を格納する |
| CD25 EB | EX DE,HL | | |
| CD26 E1 | POP HL | | |
| CD27 73 | LD (HL),E | | |
| CD28 210500 | LD HL,5 | } | (VECTOR+1)(VECTOR)に 格納された内容によって指定するアドレ スに4を加えたアドレスに0を格納する |
| CD2B EB | EX DE,HL | | |
| CD2C 2A30ED | LD HL,(VECTOR) | | |

| | | | |
|-------------|--------|----------------|---------------------|
| CD2F 19 | | ADD HL,DE | (VECTOR+1)(VECTOR)に |
| CD30 E5 | | PUSH HL | 格納された内容によって指定するアドレ |
| CD31 210100 | | LD HL,1 | スに5を加えたアドレスに1を格納する |
| CD34 EB | | EX DE,HL | |
| CD35 E1 | | POP HL | |
| CD36 73 | | LD (HL),E | |
| CD37 210600 | | LD HL,6 | |
| CD3A EB | | EX DE,HL | |
| CD3B 2A30ED | | LD HL,(VECTOR) | |
| CD3E 19 | | ADD HL,DE | |
| CD3F E5 | | PUSH HL | |
| CD40 D5 | | PUSH DE | (VECTOR+1)(VECTOR)に |
| CD41 210100 | | LD HL,1 | 格納された内容によって指定するアドレ |
| CD44 EB | | EX DE,HL | スに6を加えたアドレスに-1を格納す |
| CD45 CD2AC1 | | CALL MINUS0 | る |
| CD48 EB | | EX DE,HL | |
| CD49 D1 | | POP DE | |
| CD4A EB | | EX DE,HL | |
| CD4B E1 | | POP HL | |
| CD4C 73 | | LD (HL),E | |
| CD4D 210700 | | LD HL,7 | |
| CD50 EB | | EX DE,HL | |
| CD51 2A30ED | | LD HL,(VECTOR) | |
| CD54 19 | | ADD HL,DE | (VECTOR+1)(VECTOR)に |
| CD55 E5 | | PUSH HL | 格納された内容によって指定するアドレ |
| CD56 210100 | | LD HL,1 | スに7を加えたアドレスに1を格納する |
| CD59 EB | | EX DE,HL | |
| CD5A E1 | | POP HL | |
| CD5B 73 | | LD (HL),E | |
| CD5C 210000 | MAIN1: | LD HL,0 | |
| CD5F 221CED | | LD (WORK4),HL | (WORK4+1)(WORK4)をルー |
| CD62 212300 | | LD HL,23H | プ・カウンタとして0から35までくり |
| CD65 EB | | EX DE,HL | 返す |
| CD66 CD69CD | | CALL MAIN2 | |
| CD69 D5 | MAIN2: | PUSH DE | |
| CD6A 2A1CED | | LD HL,(WORK4) | |
| CD6D EB | | EX DE,HL | |
| CD6E 2A38ED | | LD HL,(BOARD) | |
| CD71 19 | | ADD HL,DE | (BOARD+1)(BOARD)に格納 |
| CD72 E5 | | PUSH HL | された内容にループ・カウンタを加えて |
| CD73 210000 | | LD HL,0 | 指定するアドレスに0を格納する |
| CD76 EB | | EX DE,HL | |
| CD77 E1 | | POP HL | |
| CD78 73 | | LD (HL),E | |
| CD79 2A1CED | | LD HL,(WORK4) | |
| CD7C EB | | EX DE,HL | |
| CD7D 210100 | | LD HL,1 | |
| CD80 19 | | ADD HL,DE | |
| CD81 221CED | | LD (WORK4),HL | (WORK4+1)(WORK4)をルー |
| CD84 D1 | | POP DE | プ・カウンタとするくり返し処理の終了 |
| CD85 D5 | | PUSH DE | をチェックする |

| | | | | | |
|------|----------|--------|------|-----------------------------|-------------------|
| CD86 | CD1EC1 | | CALL | ISUB0 | |
| CD89 | D1 | | POP | DE | |
| CD8A | E1 | | POP | HL | |
| CD8B | FA90CD | | JP | M,MAIN3 | |
| CD8E | E5 | | PUSH | HL | |
| CD8F | E9 | | JP | (HL) | |
| ; | | | | | |
| CD90 | 210C00 | MAIN3: | LD | HL,0CH | |
| CD93 | EB | | EX | DE,HL | } スクリーン・クリアを行う |
| CD94 | 7B | | LD | A,E | |
| CD95 | CD03C1 | | CALL | CHAR0 | |
| CD98 | 211B00 | | LD | HL,1BH | } 27個の空白を表示する |
| CD9B | EB | | EX | DE,HL | |
| CD9C | CD06C1 | | CALL | TAB0 | |
| CD9F | C3ACCD | | JP | MAIN4 | |
| ; | | | | | |
| CDA2 | 2A2A2A2A | MSG5: | DB | ***** | |
| CDA6 | 2A2A2A2A | | | | |
| CDAA | 2A2A | | | | |
| ; | | | | | |
| CDAC | 060A | MAIN4: | LD | B,10 | } 10個のアスタリスクを表示する |
| CDAE | 21A2CD | | LD | HL,MSG5 | |
| CDB1 | 7E | MAIN5: | LD | A,(HL) | |
| CDB2 | CD39C1 | | CALL | OTCHR0 | |
| CDB5 | 23 | | INC | HL | |
| CDB6 | 10F9 | | DJNZ | MAIN5 | |
| ; | | | | | |
| CDB8 | CD09C1 | | CALL | CRLF0 | } 改行する |
| CDBB | 211B00 | | LD | HL,1BH | |
| CDBE | EB | | EX | DE,HL | } 27個の空白を表示する |
| CDBF | CD06C1 | | CALL | TAB0 | |
| CDC2 | C3CFCD | | JP | MAIN6 | |
| ; | | | | | |
| CDC5 | 2A2A2032 | MSG6: | DB | '** 2D4M **' | |
| CDC9 | 44344D20 | | | | |
| CDCD | 2A2A | | | | |
| ; | | | | | |
| CDCF | 060A | MAIN6: | LD | B,10 | } タイトルを表示する |
| CDD1 | 21C5CD | | LD | HL,MSG6 | |
| CDD4 | 7E | MAIN7: | LD | A,(HL) | |
| CDD5 | CD39C1 | | CALL | OTCHR0 | |
| CDD8 | 23 | | INC | HL | |
| CDD9 | 10F9 | | DJNZ | MAIN7 | |
| ; | | | | | |
| CDDB | CD09C1 | | CALL | CRLF0 | } 改行する |
| CDDE | C306CE | | JP | MAIN8 | |
| ; | | | | | |
| CDE1 | 205B315D | MSG7: | DB | ' [1] [2] [3] [4] [5] [6] ' | |
| CDE5 | 205B325D | | | | |
| CDE9 | 205B335D | | | | |
| COED | 205B345D | | | | |

| | | | | | |
|------|----------|---------|---------|------------------------|---|
| CDF1 | 205B355D | | | | |
| CDF5 | 205B365D | | | | |
| CDF9 | 202020 | | | | ゲーム・ボードを表示する I |
| CDFC | 2A2A2A2A | DB | '*****' | | |
| CE00 | 2A2A2A2A | | | | |
| CE04 | 2A2A | | | | |
| | | ; | | | |
| CE06 | 0625 | MAIN8: | LD | B,37 | |
| CE08 | 21E1CD | | LD | HL,MSG7 | |
| CE0B | 7E | MAIN9: | LD | A,(HL) | |
| CE0C | CD39C1 | | CALL | OTCHRO | |
| CE0F | 23 | | INC | HL | |
| CE10 | 10F9 | | DJNZ | MAIN9 | |
| | | ; | | | |
| CE12 | CD09C1 | | CALL | CRLF0 | 2回改行する |
| CE15 | CD09C1 | | CALL | CRLF0 | |
| CE18 | C334CE | | JP | MAIN10 | |
| | | ; | | | |
| CE1B | 87878787 | MSG8: | DB | '████████████████████' | |
| CE1F | 87878787 | | | | |
| CE23 | 87878787 | | | | |
| CE27 | 87878787 | | | | |
| CE2B | 87878787 | | | | |
| CE2F | 87878787 | | | | ゲーム・ボードを表示する II |
| CE33 | 87 | | | | |
| | | ; | | | |
| CE34 | 0619 | MAIN10: | LD | B,25 | |
| CE36 | 211BCE | | LD | HL,MSG8 | |
| CE39 | 7E | MAIN11: | LD | A,(HL) | |
| CE3A | CD39C1 | | CALL | OTCHRO | |
| CE3D | 23 | | INC | HL | |
| CE3E | 10F9 | | DJNZ | MAIN11 | |
| | | ; | | | |
| CE40 | CD09C1 | | CALL | CRLF0 | 改行する |
| CE43 | 210100 | | LD | HL,1 | |
| CE46 | 221CED | | LD | (WORK4),HL | (WORK4+1)(WORK4)をループ・カウンタとして1から18までくり返す |
| CE49 | 211200 | | LD | HL,12H | |
| CE4C | EB | | EX | DE,HL | |
| CE4D | CD50CE | | CALL | MAIN12 | |
| CE50 | D5 | MAIN12: | PUSH | DE | |
| CE51 | C36DCE | | JP | MAIN13 | |
| | | ; | | | |
| CE54 | 87202020 | MSG9: | DB | '■■■■■■■■■■■■■■■■' | |
| CE58 | 87202020 | | | | |
| CE5C | 87202020 | | | | |
| CE60 | 87202020 | | | | |
| CE64 | 87202020 | | | | ゲーム・ボードを表示する III |
| CE68 | 87202020 | | | | |
| CE6C | 87 | | | | |
| | | ; | | | |
| CE6D | 0619 | MAIN13: | LD | B,25 | |

| | | | | | |
|------|----------|---------|------|------------|--|
| CE6F | 2154CE | | LD | HL,MSG9 | |
| CE72 | 7E | MAIN14: | LD | A,(HL) | |
| CE73 | CD39C1 | | CALL | OTCHRO | |
| CE76 | 23 | | INC | HL | |
| CE77 | 10F9 | | DJNZ | MAIN14 | |
| | | ; | | | |
| CE79 | CD09C1 | | CALL | CRLF0 | 改行する |
| CE7C | 2A1CED | | LD | HL,(WORK4) | |
| CE7F | EB | | EX | DE,HL | |
| CE80 | 210100 | | LD | HL,1 | |
| CE83 | 19 | | ADD | HL,DE | |
| CE84 | 221CED | | LD | (WORK4),HL | |
| CE87 | D1 | | POP | DE | (WORK4+1)(WORK4)をループ・カウンタとするくり返し処理の終了をチェックする |
| CE88 | D5 | | PUSH | DE | |
| CE89 | CD1EC1 | | CALL | ISUB0 | |
| CE8C | D1 | | POP | DE | |
| CE8D | E1 | | POP | HL | |
| CE8E | FA93CE | | JP | M,MAIN15 | |
| CE91 | E5 | | PUSH | HL | |
| CE92 | E9 | | JP | (HL) | |
| | | ; | | | |
| CE93 | C3AFCE | MAIN15: | JP | MAIN16 | |
| | | ; | | | |
| CE96 | 87868686 | MSG10: | DB | | |
| CE9A | 87868686 | | | | |
| CE9E | 87868686 | | | | |
| CEA2 | 87868686 | | | | |
| CEA6 | 87868686 | | | | |
| CEAA | 87868686 | | | | |
| CEAE | 87 | | | | ゲーム・ボードを表示するIV |
| | | ; | | | |
| CEAF | 0619 | MAIN16: | LD | B,25 | |
| CEB1 | 2196CE | | LD | HL,MSG10 | |
| CEB4 | 7E | MAIN17: | LD | A,(HL) | |
| CEB5 | CD39C1 | | CALL | OTCHRO | |
| CEB8 | 23 | | INC | HL | |
| CEB9 | 10F9 | | DJNZ | MAIN17 | |
| | | ; | | | |
| CEBB | 210100 | | LD | HL,1 | |
| CEBE | EB | | EX | DE,HL | |
| CEBF | 2A36ED | | LD | HL,(PCUS) | |
| CEC2 | 19 | | ADD | HL,DE | |
| CEC3 | E5 | | PUSH | HL | |
| CEC4 | 211C00 | | LD | HL,1CH | |
| CEC7 | EB | | EX | DE,HL | |
| CEC8 | E1 | | POP | HL | |
| CEC9 | 73 | | LD | (HL),E | |
| CECA | 210000 | | LD | HL,0 | カーソルを座標(28,6)に移動する |
| CECD | EB | | EX | DE,HL | |
| CECE | 2A36ED | | LD | HL,(PCUS) | |
| CED1 | 19 | | ADD | HL,DE | |

| | | | |
|---------------|-----------|---------------|-------------------------------------|
| CED2 E5 | | PUSH HL | |
| CED3 210600 | | LD HL,6 | |
| CED6 EB | | EX DE,HL | |
| CED7 E1 | | POP HL | |
| CED8 73 | | LD (HL),E | |
| CED9 C3DFCE | | JP MAIN18 | |
| | ; | | |
| CEDC C3D6D0 | MSG11: DB | 'テヨミ' | |
| | ; | | |
| CEDF 0603 | MAIN18:LD | B,3 | |
| CEE1 21DCCE | | LD HL,MSG11 | メッセージを表示する |
| CEE4 7E | MAIN19:LD | A,(HL) | |
| CEE5 CD39C1 | | CALL OTCHRO | |
| CEE8 23 | | INC HL | |
| CEE9 10F9 | | DJNZ MAIN19 | |
| | ; | | |
| CEEB 210100 | MAIN20:LD | HL,1 | |
| CEEE EB | | EX DE,HL | |
| CEEF 2A36ED | | LD HL,(PCUS) | |
| CEF2 19 | | ADD HL,DE | |
| CEF3 E5 | | PUSH HL | |
| CEF4 211E00 | | LD HL,1EH | |
| CEF7 EB | | EX DE,HL | |
| CEF8 E1 | | POP HL | |
| CEF9 73 | | LD (HL),E | |
| CEFA 210000 | | LD HL,0 | カーソルを座標(30, 8)に移動する |
| CEFD EB | | EX DE,HL | |
| CEFE 2A36ED | | LD HL,(PCUS) | |
| CF01 19 | | ADD HL,DE | |
| CF02 E5 | | PUSH HL | |
| CF03 210800 | | LD HL,8 | |
| CF06 EB | | EX DE,HL | |
| CF07 E1 | | POP HL | |
| CF08 73 | | LD (HL),E | |
| CF09 C313CF | | JP MAIN21 | |
| | ; | | |
| CF0C 28332D39 | MSG12: DB | '(3-9)?' | |
| CF10 293F20 | | | |
| | ; | | |
| CF13 0607 | MAIN21:LD | B,7 | |
| CF15 210CCF | | LD HL,MSG12 | メッセージを表示する |
| CF18 7E | MAIN22:LD | A,(HL) | |
| CF19 CD39C1 | | CALL OTCHRO | |
| CF1C 23 | | INC HL | |
| CF1D 10F9 | | DJNZ MAIN22 | |
| | ; | | |
| CF1F CD67CC | | CALL BELL | 入力を促すためブザーを鳴らす |
| CF22 CD12C1 | | CALL INKEYO | キーボードからの1文字入力を行う |
| CF25 6F | | LD L,A | |
| CF26 2600 | | LD H,0 | |
| CF28 222AED | | LD (RDEEP),HL | 入力したキャラクタ・コードを(RDEEP+1)(RDEEP)に格納する |

| | | | | | | | |
|------|--------|-----------|------------|---|---|--|--|
| CF2B | 2A2AED | LD | HL,(RDEEP) | } | ストップ・キーの場合にはシステム・モニタへジャンプする | | |
| CF2E | EB | EX | DE,HL | | | | |
| CF2F | 210300 | LD | HL,3 | | | | |
| CF32 | CD1EC1 | CALL | ISUB0 | | | | |
| CF35 | B3 | OR | E | | | | |
| CF36 | C23CCF | JP | NZ,MAIN23 | | | | |
| CF39 | C3665C | JP | MONHOT | | | | |
| ; | | | | | | | |
| CF3C | 2A2AED | MAIN23:LD | HL,(RDEEP) | } | 今入力した(RDEEP+1)(RDEEP)から30Hを減じる | | |
| CF3F | EB | EX | DE,HL | | | | |
| CF40 | 213000 | LD | HL,30H | | | | |
| CF43 | CD1EC1 | CALL | ISUB0 | | | | |
| CF46 | EB | EX | DE,HL | | | | |
| CF47 | 222AED | LD | (RDEEP),HL | | | | |
| CF4A | D5 | PUSH | DE | | | | |
| CF4B | 2A2AED | LD | HL,(RDEEP) | | | | |
| CF4E | EB | EX | DE,HL | | | | |
| CF4F | 210300 | LD | HL,3 | | | | |
| CF52 | CD1EC1 | CALL | ISUB0 | | | | |
| CF55 | 110000 | LD | DE,0 | | | | |
| CF58 | F25CCF | JP | P,MAIN24 | | | | |
| CF5B | 1C | INC | E | | | | |
| CF5C | EB | MAIN24:EX | DE,HL | } | (RDEEP+1)(RDEEP)が3~9の範囲にない場合には再度入力をくり返す | | |
| CF5D | D1 | POP | DE | | | | |
| CF5E | EB | EX | DE,HL | | | | |
| CF5F | D5 | PUSH | DE | | | | |
| CF60 | 2A2AED | LD | HL,(RDEEP) | | | | |
| CF63 | EB | EX | DE,HL | | | | |
| CF64 | 210900 | LD | HL,9 | | | | |
| CF67 | EB | EX | DE,HL | | | | |
| CF68 | CD1EC1 | CALL | ISUB0 | | | | |
| CF6B | 110000 | LD | DE,0 | | | | |
| CF6E | F272CF | JP | P,MAIN25 | | | | |
| CF71 | 1C | INC | E | | | | |
| CF72 | EB | MAIN25:EX | DE,HL | | | | |
| CF73 | D1 | POP | DE | | | | |
| CF74 | 19 | ADD | HL,DE | } | (RDEEP+1)(RDEEP)を8倍して16を減じる | | |
| CF75 | EB | EX | DE,HL | | | | |
| CF76 | 7B | LD | A,E | | | | |
| CF77 | B2 | OR | D | | | | |
| CF78 | CA7ECF | JP | Z,MAIN26 | | | | |
| CF7B | C3EBCE | JP | MAIN20 | | | | |
| ; | | | | | | | |
| CF7E | 2A2AED | MAIN26:LD | HL,(RDEEP) | | | | |
| CF81 | EB | EX | DE,HL | | | | |
| CF82 | 210800 | LD | HL,8 | | | | |
| CF85 | CD21C1 | CALL | IMULT0 | | | | |
| CF88 | 211000 | LD | HL,10H | | | | |
| CF8B | CD1EC1 | CALL | ISUB0 | | | | |
| CF8E | EB | EX | DE,HL | | | | |
| CF8F | 222AED | LD | (RDEEP),HL | | | | |

| | | | | | |
|------|----------|-----------|---------------------|---|----------------------|
| CF92 | 210100 | LD | HL,1 | } | カーソルを座標(28, 10)に移動する |
| CF95 | EB | EX | DE,HL | | |
| CF96 | 2A36ED | LD | HL,(PCUS) | | |
| CF99 | 19 | ADD | HL,DE | | |
| CF9A | E5 | PUSH | HL | | |
| CF9B | 211C00 | LD | HL,1CH | | |
| CF9E | EB | EX | DE,HL | | |
| CF9F | E1 | POP | HL | | |
| CFA0 | 73 | LD | (HL),E | | |
| CFA1 | 210000 | LD | HL,0 | | |
| CFA4 | EB | EX | DE,HL | | |
| CFA5 | 2A36ED | LD | HL,(PCUS) | | |
| CFA8 | 19 | ADD | HL,DE | | |
| CFA9 | E5 | PUSH | HL | | |
| CFAA | 210A00 | LD | HL,0AH | | |
| CFAD | EB | EX | DE,HL | | |
| CFAE | E1 | POP | HL | | |
| CFAF | 73 | LD | (HL),E | | |
| CFB0 | C3B8CF | JP | MAIN27 | | |
| CFB3 | C6DDBDDE | ; | MSG13: DB 'ニンスウ' | } | メッセージを表示する |
| CFB7 | B3 | | | | |
| CFB8 | 0605 | MAIN27:LD | B,5 | | |
| CFBA | 21B3CF | | LD HL,MSG13 | | |
| CFBD | 7E | MAIN28:LD | A,(HL) | | |
| CFBE | CD39C1 | | CALL OTCHRO | | |
| CFC1 | 23 | | INC HL | } | カーソルを座標(30, 12)に移動する |
| CFC2 | 10F9 | | DJNZ MAIN28 | | |
| CFC4 | 210100 | ; | MAIN29:LD HL,1 | | |
| CFC7 | EB | | EX DE,HL | | |
| CFC8 | 2A36ED | | LD HL,(PCUS) | | |
| CFCB | 19 | | ADD HL,DE | | |
| CFCC | E5 | | PUSH HL | | |
| CFCD | 211E00 | | LD HL,1EH | | |
| CFD0 | EB | | EX DE,HL | | |
| CFD1 | E1 | | POP HL | | |
| CFD2 | 73 | | LD (HL),E | | |
| CFD3 | 210000 | | LD HL,0 | | |
| CFD6 | EB | | EX DE,HL | | |
| CFD7 | 2A36ED | | LD HL,(PCUS) | | |
| CFDA | 19 | | ADD HL,DE | | |
| CFDB | E5 | | PUSH HL | | |
| CFDC | 210C00 | | LD HL,0CH | | |
| CFDF | EB | | EX DE,HL | | |
| CFE0 | E1 | | POP HL | | |
| CFE1 | 73 | | LD (HL),E | | |
| CFE2 | C3ECCF | | JP MAIN30 | | |
| CFE5 | 28302D32 | ; | MSG14: DB '(0-2)? ' | | |

CFE9 293F20

CFEC 0607

CFEE 21E5CF

CFF1 7E

CFF2 CD39C1

CFF5 23

CFF6 10F9

;

MAIN30:LD B,7
LD HL,MSG14

MAIN31:LD A,(HL)

CALL OTCHRO

INC HL

DJNZ MAIN31

メッセージを表示する

;

CFF8 CD67CC

CFFB CD12C1

CFFE 6F

CFFF 2600

D001 220AED

D004 2A0AED

D007 EB

D008 210300

D00B CD1EC1

D00E B3

D00F C215D0

D012 C3665C

CALL BELL

CALL INKEY0

LD L,A

LD H,0

LD (PLAYER),HL

LD HL,(PLAYER)

EX DE,HL

LD HL,3

CALL ISUB0

OR E

JP NZ,MAIN32

JP MONHOT

入力を促すためブザーを鳴らす

キーボードからの1文字入力を行う

入力したキャラクタ・コードを (PLAYER+1) (PLAYER) に格納する

ストップ・キーの場合にはシステム・モニタへジャンプする

;

MAIN32:LD HL,(PLAYER)

EX DE,HL

LD HL,30H

CALL ISUB0

EX DE,HL

LD (PLAYER),HL

PUSH DE

LD HL,(PLAYER)

EX DE,HL

LD HL,0

CALL ISUB0

LD DE,0

JP P,MAIN33

INC E

MAIN33:EX DE,HL

POP DE

EX DE,HL

PUSH DE

LD HL,(PLAYER)

EX DE,HL

LD HL,2

EX DE,HL

CALL ISUB0

LD DE,0

JP P,MAIN34

INC E

MAIN34:EX DE,HL

POP DE

ADD HL,DE

(PLAYER+1) (PLAYER) に格納されたキャラクタ・コードから30Hを減じてバイナリ・コードに変換する

(PLAYER+1) (PLAYER) が0~2の範囲にない場合には再度入力を行う

| | | | | | | | |
|------|--------|-----------|-------------|---|---------------------------------|--|--|
| D04E | EB | EX | DE,HL | } | | | |
| D04F | 7B | LD | A,E | | | | |
| D050 | B2 | OR | D | | | | |
| D051 | CA57D0 | JP | Z,MAIN35 | | | | |
| D054 | C3C4CF | JP | MAIN29 | | | | |
| ; | | | | | | | |
| D057 | 210100 | MAIN35:LD | HL,1 | } | (STONE+1)(STONE)に1を | | |
| D05A | 2206ED | LD | (SOTNE),HL | | 格納する | | |
| D05D | 2A0AED | LD | HL,(PLAYER) | } | 人間対コンピュータの対戦でなければつ ぎの処理を省略する | | |
| D060 | EB | EX | DE,HL | | | | |
| D061 | 210100 | LD | HL,1 | | | | |
| D064 | CD1EC1 | CALL | ISUB0 | | | | |
| D067 | B3 | OR | E | | | | |
| D068 | CA6ED0 | JP | Z,MAIN36 | | | | |
| D06B | C320D1 | JP | MAIN45 | | | | |
| ; | | | | | | | |
| D06E | 210100 | MAIN36:LD | HL,1 | } | カーソルを座標(28, 14)に移動する | | |
| D071 | EB | EX | DE,HL | | | | |
| D072 | 2A36ED | LD | HL,(PCUS) | | | | |
| D075 | 19 | ADD | HL,DE | | | | |
| D076 | E5 | PUSH | HL | | | | |
| D077 | 211C00 | LD | HL,1CH | | | | |
| D07A | EB | EX | DE,HL | | | | |
| D07B | E1 | POP | HL | | | | |
| D07C | 73 | LD | (HL),E | | | | |
| D07D | 210000 | LD | HL,0 | | | | |
| D080 | EB | EX | DE,HL | | | | |
| D081 | 2A36ED | LD | HL,(PCUS) | | | | |
| D084 | 19 | ADD | HL,DE | | | | |
| D085 | E5 | PUSH | HL | | | | |
| D086 | 210E00 | LD | HL,0EH | | | | |
| D089 | EB | EX | DE,HL | | | | |
| D08A | E1 | POP | HL | | | | |
| D08B | 73 | LD | (HL),E | | | | |
| D08C | C392D0 | JP | MAIN37 | | | | |
| ; | | | | | | | |
| D08F | BEDDC3 | MSG15: DB | ‘センテ’ | } | メッセージを表示する | | |
| ; | | | | | | | |
| D092 | 0603 | MAIN37:LD | B,3 | | | | |
| D094 | 218FD0 | LD | HL,MSG15 | | | | |
| D097 | 7E | MAIN38:LD | A,(HL) | | | | |
| D098 | CD39C1 | CALL | OTCHRO | | | | |
| D09B | 23 | INC | HL | | | | |
| D09C | 10F9 | DJNZ | MAIN38 | | | | |
| ; | | | | | | | |
| D09E | 210100 | MAIN39:LD | HL,1 | } | | | |
| D0A1 | EB | EX | DE,HL | | | | |
| D0A2 | 2A36ED | LD | HL,(PCUS) | | | | |
| D0A5 | 19 | ADD | HL,DE | | | | |
| D0A6 | E5 | PUSH | HL | | | | |
| D0A7 | 211E00 | LD | HL,1EH | | | | |

| | | | | |
|------|----------|-----------|------------|---|
| DOAA | EB | EX | DE,HL | カーソルを座標(30, 16)に移動する |
| DOAB | E1 | POP | HL | |
| DOAC | 73 | LD | (HL),E | |
| DOAD | 210000 | LD | HL,0 | |
| DOBO | EB | EX | DE,HL | |
| DOB1 | 2A36ED | LD | HL,(PCUS) | |
| DOB4 | 19 | ADD | HL,DE | |
| DOB5 | E5 | PUSH | HL | |
| DOB6 | 211000 | LD | HL,10H | |
| DOB9 | EB | EX | DE,HL | |
| DOBA | E1 | POP | HL | メッセージを表示する |
| DOBB | 73 | LD | (HL),E | |
| DOBC | C3C6D0 | JP | MAIN40 | |
| ; | | | | |
| DOBF | 28592F4E | MSG16: DB | (Y/N)? | |
| DOC3 | 293F20 | | | |
| ; | | | | |
| DOC6 | 0607 | MAIN40:LD | B,7 | |
| DOC8 | 21BFD0 | LD | HL,MSG16 | |
| DOCB | 7E | MAIN41:LD | A,(HL) | |
| DOCC | CD39C1 | CALL | OTCHRO | |
| DOCF | 23 | INC | HL | 入力を促すためブザーを鳴らす キーボードからの1文字入力を行う 入力したキャラクタ・コードを(WORK4+1)(WORK4)に格納する |
| DOD0 | 10F9 | DJNZ | MAIN41 | |
| ; | | | | |
| DOD2 | CD67CC | CALL | BELL | |
| DOD5 | CD12C1 | CALL | INKEY0 | |
| DOD8 | 6F | LD | L,A | |
| DOD9 | 2600 | LD | H,0 | |
| DODB | 221CED | LD | (WORK4),HL | |
| DODE | 2A1CED | LD | HL,(WORK4) | |
| DOE1 | EB | EX | DE,HL | |
| DOE2 | 210300 | LD | HL,3 | |
| DOE5 | CD1EC1 | CALL | ISUB0 | |
| DOE8 | B3 | OR | E | |
| DOE9 | C2EFD0 | JP | NZ,MAIN42 | |
| DOEC | C3665C | JP | MONHOT | |
| ; | | | | |
| DOEF | 2A1CED | MAIN42:LD | HL,(WORK4) | |
| DOF2 | EB | EX | DE,HL | |
| DOF3 | 215900 | LD | HL,59H | |
| DOF6 | CD1EC1 | CALL | ISUB0 | |
| DOF9 | B3 | OR | E | 先手の場合には(SOTNE+1)(SOTNE)に2を格納する |
| DOFA | C206D1 | JP | NZ,MAIN43 | |
| DOFD | 210200 | LD | HL,2 | |
| D100 | 2206ED | LD | (SOTNE),HL | |
| D103 | C320D1 | JP | MAIN45 | |
| ; | | | | |
| D106 | 2A1CED | MAIN43:LD | HL,(WORK4) | |
| D109 | EB | EX | DE,HL | |
| D10A | 214E00 | LD | HL,4EH | |
| D10D | CD1EC1 | CALL | ISUB0 | |

| | | | | |
|------|--------|-----------|-------------|---|
| D110 | B3 | OR | E | } 後手の場合には(SOTNE+1)(SOTNE)に1を格納する |
| D111 | C21DD1 | JP | NZ,MAIN44 | |
| D114 | 210100 | LD | HL,1 | |
| D117 | 2206ED | LD | (SOTNE),HL | |
| D11A | C320D1 | JP | MAIN45 | |
| ; | | | | |
| D11D | C39ED0 | MAIN44:JP | MAIN39 | } 先手でも後手でもない場合には再度入力をくり返す |
| ; | | | | |
| D120 | 210000 | MAIN45:LD | HL,0 | } (PLAYS+1)(PLAYS)に0を格納する |
| D123 | 2208ED | LD | (PLAYS),HL | |
| D126 | 210300 | MAIN46:LD | HL,3 | } (SOTNE+1)(SOTNE)が1であれば2に変換し2であれば1に変換する |
| D129 | EB | EX | DE,HL | |
| D12A | 2A06ED | LD | HL,(SOTNE) | |
| D12D | CD1EC1 | CALL | ISUB0 | |
| D130 | EB | EX | DE,HL | |
| D131 | 2206ED | LD | (SOTNE),HL | } (PLAYS+1)(PLAYS)に1を加える |
| D134 | 2A08ED | LD | HL,(PLAYS) | |
| D137 | EB | EX | DE,HL | |
| D138 | 210100 | LD | HL,1 | |
| D13B | 19 | ADD | HL,DE | |
| D13C | 2208ED | LD | (PLAYS),HL | } |
| D13F | D5 | PUSH | DE | |
| D140 | 2A0AED | LD | HL,(PLAYER) | |
| D143 | EB | EX | DE,HL | |
| D144 | 210100 | LD | HL,1 | |
| D147 | CD1EC1 | CALL | ISUB0 | } |
| D14A | B3 | OR | E | |
| D14B | 110000 | LD | DE,0 | |
| D14E | C252D1 | JP | NZ,MAIN47 | |
| D151 | 1C | INC | E | |
| D152 | EB | MAIN47:EX | DE,HL | } |
| D153 | D1 | POP | DE | |
| D154 | EB | EX | DE,HL | |
| D155 | D5 | PUSH | DE | |
| D156 | 2A06ED | LD | HL,(SOTNE) | |
| D159 | EB | EX | DE,HL | } |
| D15A | 210200 | LD | HL,2 | |
| D15D | CD1EC1 | CALL | ISUB0 | |
| D160 | B3 | OR | E | |
| D161 | 110000 | LD | DE,0 | |
| D164 | C268D1 | JP | NZ,MAIN48 | } コンピュータの手番であればコンピュータの思考ルーチン『COMP』へジャンプする |
| D167 | 1C | INC | E | |
| D168 | EB | MAIN48:EX | DE,HL | |
| D169 | D1 | POP | DE | |
| D16A | CD21C1 | CALL | IMULT0 | |
| D16D | D5 | PUSH | DE | } |
| D16E | 2A0AED | LD | HL,(PLAYER) | |
| D171 | EB | EX | DE,HL | |
| D172 | 210000 | LD | HL,0 | |
| D175 | CD1EC1 | CALL | ISUB0 | |
| D178 | B3 | OR | E | |

| | | | | |
|------|----------|-------------|------------|----------------------------|
| D179 | 110000 | LD | DE,0 | |
| D17C | C280D1 | JP | NZ,MAIN49 | |
| D17F | 1C | INC | E | |
| D180 | EB | MAIN49:EX | DE,HL | |
| D181 | D1 | POP | DE | |
| D182 | 19 | ADD | HL,DE | |
| D183 | EB | EX | DE,HL | |
| D184 | 7B | LD | A,E | |
| D185 | B2 | OR | D | |
| D186 | CA8CD1 | JP | Z,MAIN50 | |
| D189 | C30CD3 | JP | COMP | |
| ; | | | | |
| D18C | 211C00 | MAIN50:LD | HL,1CH | } (DEEP+1)(DEEP)に28を格納する |
| D18F | 2216ED | LD | (DEEP),HL | |
| D192 | 211300 | LD | HL,13H | } (DEEP1+1)(DEEP1)に19を格納する |
| D195 | 2218ED | LD | (DEEP1),HL | |
| D198 | CD46CB | CALL | DSUB | } 座標(28, 19)から石を表示する |
| D19B | CD2ACC | MAIN51:CALL | DERA | |
| D19E | 210100 | LD | HL,1 | } メッセージを消去する |
| D1A1 | EB | EX | DE,HL | |
| D1A2 | 2A36ED | LD | HL,(PCUS) | |
| D1A5 | 19 | ADD | HL,DE | |
| D1A6 | E5 | PUSH | HL | |
| D1A7 | 211E00 | LD | HL,1EH | |
| D1AA | EB | EX | DE,HL | |
| D1AB | E1 | POP | HL | |
| D1AC | 73 | LD | (HL),E | |
| D1AD | 210000 | LD | HL,0 | |
| D1B0 | EB | EX | DE,HL | } カーソルを座標(30, 23)に移動する |
| D1B1 | 2A36ED | LD | HL,(PCUS) | |
| D1B4 | 19 | ADD | HL,DE | |
| D1B5 | E5 | PUSH | HL | |
| D1B6 | 211700 | LD | HL,17H | |
| D1B9 | EB | EX | DE,HL | |
| D1BA | E1 | POP | HL | |
| D1BB | 73 | LD | (HL),E | |
| D1BC | C3C6D1 | JP | MAIN52 | |
| ; | | | | |
| D1BF | 28312D36 | MSG17: DB | '(1-6)?' | |
| D1C3 | 293F20 | | | |
| ; | | | | |
| D1C6 | 0607 | MAIN52:LD | B,7 | } メッセージを表示する |
| D1C8 | 21BFD1 | LD | HL,MSG17 | |
| D1CB | 7E | MAIN53:LD | A,(HL) | |
| D1CC | CD39C1 | CALL | OTCHR0 | |
| D1CF | 23 | INC | HL | |
| D1D0 | 10F9 | DJNZ | MAIN53 | |
| ; | | | | |
| D1D2 | CD67CC | CALL | BELL | } 入力を促すためブザーを鳴らす |
| D1D5 | CD12C1 | CALL | INKEY0 | |
| D1D8 | 6F | LD | L,A | } キーボードからの1文字入力を行う |
| | | | | |

| | | | | | |
|------|--------|-----------|------------|---|-------------------------------------|
| D1D9 | 2600 | LD | H,0 | } | 入力したキャラクタ・コードを(COORD+1)(COORD)に格納する |
| D1DB | 2234ED | LD | (COORD),HL | | |
| D1DE | 2A34ED | LD | HL,(COORD) | } | ストップ・キーの場合にはシステム・モニタへジャンプする |
| D1E1 | EB | EX | DE,HL | | |
| D1E2 | 210300 | LD | HL,3 | | |
| D1E5 | CD1EC1 | CALL | ISUB0 | | |
| D1E8 | B3 | OR | E | | |
| D1E9 | C2EFD1 | JP | NZ,MAIN54 | | |
| D1EC | C3665C | JP | MONHOT | | |
| ; | | | | | |
| D1EF | 2A34ED | MAIN54:LD | HL,(COORD) | } | 入力したキャラクタ・コードから31Hを減じてバイナリ・コードに変換する |
| D1F2 | EB | EX | DE,HL | | |
| D1F3 | 213100 | LD | HL,31H | | |
| D1F6 | CD1EC1 | CALL | ISUB0 | | |
| D1F9 | EB | EX | DE,HL | | |
| D1FA | 2234ED | LD | (COORD),HL | | |
| D1FD | D5 | PUSH | DE | | |
| D1FE | 2A34ED | LD | HL,(COORD) | | |
| D201 | EB | EX | DE,HL | | |
| D202 | 210000 | LD | HL,0 | | |
| D205 | CD1EC1 | CALL | ISUB0 | | |
| D208 | 110000 | LD | DE,0 | | |
| D20B | F20FD2 | JP | P,MAIN55 | | |
| D20E | 1C | INC | E | | |
| D20F | EB | MAIN55:EX | DE,HL | | |
| D210 | D1 | POP | DE | | |
| D211 | EB | EX | DE,HL | | |
| D212 | D5 | PUSH | DE | | |
| D213 | 2A34ED | LD | HL,(COORD) | | |
| D216 | EB | EX | DE,HL | | |
| D217 | 210500 | LD | HL,5 | | |
| D21A | EB | EX | DE,HL | | |
| D21B | CD1EC1 | CALL | ISUB0 | | |
| D21E | 110000 | LD | DE,0 | | |
| D221 | F225D2 | JP | P,MAIN56 | | |
| D224 | 1C | INC | E | | |
| D225 | EB | MAIN56:EX | DE,HL | | |
| D226 | D1 | POP | DE | | |
| D227 | 19 | ADD | HL,DE | | |
| D228 | EB | EX | DE,HL | | |
| D229 | D5 | PUSH | DE | | |
| D22A | D5 | PUSH | DE | | |
| D22B | 2A34ED | LD | HL,(COORD) | | |
| D22E | EB | EX | DE,HL | | |
| D22F | 2A38ED | LD | HL,(BOARD) | | |
| D232 | 19 | ADD | HL,DE | | |
| D233 | D1 | POP | DE | | |
| D234 | 6E | LD | L,(HL) | | |
| D235 | 2600 | LD | H,0 | | |
| D237 | EB | EX | DE,HL | | |
| D238 | 210000 | LD | HL,0 | | |

入力データが0～5の範囲でない場合または指定した位置が6個の石でふさがっている場合には再度入力をくり返す

| | | | | |
|------|--------|-----------|-------------|--|
| D23B | EB | EX | DE,HL | |
| D23C | CD1EC1 | CALL | ISUB0 | |
| D23F | 110000 | LD | DE,0 | |
| D242 | F246D2 | JP | P,MAIN57 | |
| D245 | 1C | INC | E | |
| D246 | EB | MAIN57:EX | DE,HL | |
| D247 | D1 | POP | DE | |
| D248 | 19 | ADD | HL,DE | |
| D249 | EB | EX | DE,HL | |
| D24A | 7B | LD | A,E | |
| D24B | B2 | OR | D | |
| D24C | CA52D2 | JP | Z,MAIN58 | |
| D24F | C39BD1 | JP | MAIN51 | |
| ; | | | | |
| D252 | 2A34ED | MAIN58:LD | HL,(COORD) | } (COORD+1)(COORD)を(COORD1+1)(COORD1)に退避する |
| D255 | 2210ED | LD | (COORD1),HL | |
| D258 | CD16CA | CALL | DISP | } 石の落下をアニメーション表示する |
| D25B | 2A06ED | LD | HL,(SOTNE) | |
| D25E | 220CED | LD | (STONE1),HL | } (SOTNE+1)(SOTNE)を(STONE1+1)(STONE1)に退避する |
| D261 | CDCEC5 | CALL | VAL | |
| D264 | D5 | PUSH | DE | } 評価値を計算する |
| D265 | 2A28ED | LD | HL,(VALUE) | |
| D268 | EB | EX | DE,HL | |
| D269 | 21C800 | LD | HL,0C8H | |
| D26C | CD1EC1 | CALL | ISUB0 | |
| D26F | 110000 | LD | DE,0 | |
| D272 | F276D2 | JP | P,MAIN59 | |
| D275 | 1C | INC | E | |
| D276 | EB | MAIN59:EX | DE,HL | |
| D277 | D1 | POP | DE | |
| D278 | EB | EX | DE,HL | |
| D279 | D5 | PUSH | DE | |
| D27A | 2A08ED | LD | HL,(PLAYS) | } いずれのプレイヤーも4個の石が並んでおらずかつ全ての石を打ち尽くしていない場合には次の手番をくり返す |
| D27D | EB | EX | DE,HL | |
| D27E | 212400 | LD | HL,24H | |
| D281 | CD1EC1 | CALL | ISUB0 | |
| D284 | 110000 | LD | DE,0 | |
| D287 | F28BD2 | JP | P,MAIN60 | |
| D28A | 1C | INC | E | |
| D28B | EB | MAIN60:EX | DE,HL | |
| D28C | D1 | POP | DE | |
| D28D | CD21C1 | CALL | IMULT0 | |
| D290 | 7B | LD | A,E | |
| D291 | B2 | OR | D | |
| D292 | CA98D2 | JP | Z,MAIN61 | |
| D295 | C326D1 | JP | MAIN46 | |
| ; | | | | |
| D298 | 211C00 | MAIN61:LD | HL,1CH | } (DEEP+1)(DEEP)に28を格納する |
| D29B | 2216ED | LD | (DEEP),HL | |
| D29E | 211300 | LD | HL,13H | } (DEEP1+1)(DEEP1)に19を格納する |
| D2A1 | 2218ED | LD | (DEEP1),HL | |

| | | | | |
|------|----------|-------------|---------------|---------------------------------------|
| D2A4 | CD46CB | CALL | DSUB | } 座標(28, 19)から石を表示する |
| D2A7 | CD2ACC | MAIN62:CALL | DERA | |
| D2AA | 210100 | LD | HL,1 | } メッセージを消去する |
| D2AD | EB | EX | DE,HL | |
| D2AE | 2A36ED | LD | HL,(PCUS) | } カースルを座標(30, 23)に移動する |
| D2B1 | 19 | ADD | HL,DE | |
| D2B2 | E5 | PUSH | HL | |
| D2B3 | 211E00 | LD | HL,1EH | |
| D2B6 | EB | EX | DE,HL | |
| D2B7 | E1 | POP | HL | |
| D2B8 | 73 | LD | (HL),E | |
| D2B9 | 210000 | LD | HL,0 | |
| D2BC | EB | EX | DE,HL | |
| D2BD | 2A36ED | LD | HL,(PCUS) | |
| D2C0 | 19 | ADD | HL,DE | |
| D2C1 | E5 | PUSH | HL | |
| D2C2 | 211700 | LD | HL,17H | } ウイニング・メッセージを表示する |
| D2C5 | EB | EX | DE,HL | |
| D2C6 | E1 | POP | HL | |
| D2C7 | 73 | LD | (HL),E | |
| D2C8 | C3D2D2 | JP | MAIN63 | |
| D2CB | C920B6C1 | ; | MSG18: DB | |
| D2CF | 212120 | | ' / カチ!! ' | |
| D2D2 | 0607 | ; | MAIN63:LD | |
| D2D4 | 21CBD2 | | B,7 | |
| D2D7 | 7E | | LD HL,MSG18 | |
| D2D8 | CD39C1 | MAIN64:LD | A,(HL) | |
| D2DB | 23 | | CALL OTCHRO | |
| D2DC | 10F9 | | INC HL | |
| | | | DJNZ MAIN64 | |
| D2DE | CD12C1 | ; | CALL INKEYO | } キーボードから1文字入力を行う |
| D2E1 | 6F | | LD L,A | |
| D2E2 | 2600 | | LD H,0 | } キャラクタ・コードを(WORK4+1) (WORK4)に格納する |
| D2E4 | 221CED | | LD (WORK4),HL | |
| D2E7 | 2A1CED | | LD HL,(WORK4) | } ストップ・キーの場合にはシステム・モニタへジャンプする |
| D2EA | EB | | EX DE,HL | |
| D2EB | 210300 | | LD HL,3 | |
| D2EE | CD1EC1 | | CALL ISUBO | |
| D2F1 | B3 | | OR E | |
| D2F2 | C2F8D2 | | JP NZ,MAIN65 | |
| D2F5 | C3665C | | JP MONHOT | } リターン・キーでなければ再度入力をくり返す |
| D2F8 | 2A1CED | ; | MAIN65:LD | |
| D2FB | EB | | HL,(WORK4) | |
| D2FC | 210D00 | | EX DE,HL | |
| D2FF | CD1EC1 | | LD HL,0DH | |
| D302 | B3 | | CALL ISUBO | |
| D303 | CA09D3 | | OR E | |
| D306 | C3A7D2 | | JP Z,MAIN66 | |
| | | | JP MAIN62 | |

| | | | | | |
|------|--------|--------|-------------------|-------------|---|
| D309 | C35CCD | ; | MAIN66:JP | MAIN1 | } リターンキーの場合には再ゲームヘジャンプする |
| | | ; | computer thinking | | |
| D30C | 2A08ED | COMP: | LD | HL,(PLAYS) | 手番が1~3の場合には0~5の乱数を発生しコンピュータの手として与える |
| D30F | EB | | EX | DE,HL | |
| D310 | 210400 | | LD | HL,4 | |
| D313 | CD1EC1 | | CALL | ISUB0 | |
| D316 | F233D3 | | JP | P,COMP1 | |
| D319 | D5 | | PUSH | DE | |
| D31A | C5 | | PUSH | BC | |
| D31B | 210600 | | LD | HL,6 | |
| D31E | EB | | EX | DE,HL | |
| D31F | CD2DC1 | | CALL | RND0 | |
| D322 | EB | | EX | DE,HL | |
| D323 | C1 | | POP | BC | |
| D324 | D1 | | POP | DE | |
| D325 | EB | | EX | DE,HL | |
| D326 | 210100 | | LD | HL,1 | |
| D329 | CD1EC1 | | CALL | ISUB0 | |
| D32C | EB | | EX | DE,HL | |
| D32D | 2234ED | | LD | (COORD),HL | |
| D330 | C352D2 | | JP | MAIN58 | |
| D333 | 210000 | COMP1: | LD | HL,0 | (DEEP+1)(DEEP)に0を格納する |
| D336 | 2216ED | | LD | (DEEP),HL | |
| D339 | 2A06ED | | LD | HL,(SOTNE) | (SOTNE+1)(SOTNE)を(STONE1+1)(STONE1)に退避する |
| D33C | 220CED | | LD | (STONE1),HL | |
| D33F | 210100 | | LD | HL,1 | (WORK4+1)(WORK4)をループ・カウンタとして1から(RDEEP+1)(RDEEP)に9を加えた値までくり返す。 |
| D342 | 221CED | | LD | (WORK4),HL | |
| D345 | 2A2AED | | LD | HL,(RDEEP) | |
| D348 | EB | | EX | DE,HL | |
| D349 | 210900 | | LD | HL,9 | |
| D34C | 19 | | ADD | HL,DE | |
| D34D | EB | | EX | DE,HL | |
| D34E | CD51D3 | | CALL | COMP2 | |
| D351 | D5 | COMP2: | PUSH | DE | |
| D352 | 2A1CED | | LD | HL,(WORK4) | |
| D355 | EB | | EX | DE,HL | (STACK+1)(STACK)の内容に(WORK4+1)(WORK4)の内容を加えて指定するアドレスに0を格納する |
| D356 | 2A32ED | | LD | HL,(STACK) | |
| D359 | 19 | | ADD | HL,DE | |
| D35A | E5 | | PUSH | HL | |
| D35B | 210000 | | LD | HL,0 | |
| D35E | EB | | EX | DE,HL | |
| D35F | E1 | | POP | HL | |
| D360 | 73 | | LD | (HL),E | |
| D361 | 2A1CED | | LD | HL,(WORK4) | |
| D364 | EB | | EX | DE,HL | |
| D365 | 210800 | | LD | HL,8 | |
| D368 | 19 | | ADD | HL,DE | |

| | | | | |
|------|--------|--------|---------------|--|
| D369 | EB | EX | DE,HL | (STACK+1)(STACK)の内容に(WORK4+1)(WORK4)の内容を加えさらに8を加えて指定するアドレスにFFHを格納する |
| D36A | 2A32ED | LD | HL,(STACK) | |
| D36D | 19 | ADD | HL,DE | |
| D36E | E5 | PUSH | HL | |
| D36F | 21FF00 | LD | HL,0FFH | |
| D372 | EB | EX | DE,HL | |
| D373 | E1 | POP | HL | |
| D374 | 73 | LD | (HL),E | |
| D375 | 2A1CED | LD | HL,(WORK4) | |
| D378 | EB | EX | DE,HL | |
| D379 | 211000 | LD | HL,10H | (WORK4+1)(WORK4)をループ・カウンタとし増分を16としてくり返し処理の終了をチェックする |
| D37C | 19 | ADD | HL,DE | |
| D37D | 221CED | LD | (WORK4),HL | |
| D380 | D1 | POP | DE | |
| D381 | D5 | PUSH | DE | |
| D382 | CD1EC1 | CALL | ISUB0 | |
| D385 | D1 | POP | DE | |
| D386 | E1 | POP | HL | |
| D387 | FA8CD3 | JP | M,COMP3 | |
| D38A | E5 | PUSH | HL | |
| D38B | E9 | JP | (HL) | |
| ; | | | | |
| D38C | 110000 | COMP3: | LD DE,0 | 次のくり返し処理のためにスタックの操作を行う |
| D38F | CD92D3 | | CALL COMP4 | |
| D392 | D5 | COMP4: | PUSH DE | |
| D393 | D5 | | PUSH DE | |
| D394 | 2A16ED | | LD HL,(DEEP) | |
| D397 | EB | | EX DE,HL | |
| D398 | 2A2AED | | LD HL,(RDEEP) | |
| D39B | EB | | EX DE,HL | |
| D39C | CD1EC1 | | CALL ISUB0 | |
| D39F | 110000 | | LD DE,0 | |
| D3A2 | F2A6D3 | | JP P,COMP5 | (RDEEP+1)(RDEEP)の内容が(DEEP+1)(DEEP)の内容未満かまたは(VALUE+1)(VALUE)の内容が1か200の場合には『COMP11』へジャンプする |
| D3A5 | 1C | | INC E | |
| D3A6 | EB | COMP5: | EX DE,HL | |
| D3A7 | D1 | | POP DE | |
| D3A8 | EB | | EX DE,HL | |
| D3A9 | D5 | | PUSH DE | |
| D3AA | 2A28ED | | LD HL,(VALUE) | |
| D3AD | EB | | EX DE,HL | |
| D3AE | 210100 | | LD HL,1 | |
| D3B1 | CD1EC1 | | CALL ISUB0 | |
| D3B4 | B3 | | OR E | COMP6: |
| D3B5 | 110000 | | LD DE,0 | |
| D3B8 | C2BCD3 | | JP NZ,COMP6 | |
| D3BB | 1C | | INC E | |
| D3BC | EB | COMP6: | EX DE,HL | |
| D3BD | D1 | | POP DE | |
| D3BE | 19 | | ADD HL,DE | |
| D3BF | EB | | EX DE,HL | |
| D3C0 | D5 | | PUSH DE | |


```

D3C1 2A28ED      LD    HL,(VALUE)
D3C4 EB          EX    DE,HL
D3C5 21C800      LD    HL,0C8H
D3C8 CD1EC1      CALL  ISUB0
D3CB B3          OR     E
D3CC 110000      LD    DE,0
D3CF C2D3D3      JP     NZ,COMP7
D3D2 1C          INC    E
D3D3 EB          COMP7: EX    DE,HL
D3D4 D1          POP    DE
D3D5 19          ADD    HL,DE
D3D6 EB          EX    DE,HL
D3D7 7B          LD    A,E
D3D8 B2          OR     D
D3D9 CADFD3      JP     Z,COMP8
D3DC C3C9D4      JP     COMP11

```

```

;
D3DF CD03C5      COMP8: CALL  LIST
D3E2 D5          COMP9: PUSH  DE
D3E3 2A16ED      LD    HL,(DEEP)
D3E6 EB          EX    DE,HL
D3E7 2A32ED      LD    HL,(STACK)
D3EA 19          ADD    HL,DE
D3EB D1          POP    DE
D3EC 6E          LD    L,(HL)
D3ED 2600        LD    H,0
D3EF EB          EX    DE,HL
D3F0 210300      LD    HL,3
D3F3 CD1EC1      CALL  ISUB0
D3F6 F2FCD3      JP     P,COMP10
D3F9 C3C9D4      JP     COMP11

```

} 可能な次の手をワーク・エリアに求める

(STACK+1)(STACK)の内容
に(DEEP+1)(DEEP)の内容を
加えて指定するアドレスの内容が3未満
の場合『COMP11』へジャンプする

```

;
D3FC 2A16ED      COMP10: LD    HL,(DEEP)
D3FF EB          EX    DE,HL
D400 2A32ED      LD    HL,(STACK)
D403 19          ADD    HL,DE
D404 E5          PUSH  HL
D405 D5          PUSH  DE
D406 2A16ED      LD    HL,(DEEP)
D409 EB          EX    DE,HL
D40A 2A32ED      LD    HL,(STACK)
D40D 19          ADD    HL,DE
D40E D1          POP    DE
D40F 6E          LD    L,(HL)
D410 2600        LD    H,0
D412 EB          EX    DE,HL
D413 210100      LD    HL,1
D416 CD1EC1      CALL  ISUB0
D419 E1          POP    HL
D41A 73          LD    (HL),E
D41B D5          PUSH  DE

```

(STACK+1)(STACK)の内容
に(DEEP+1)(DEEP)の内容を
加えて指定するアドレスの内容から1を
減じる

| | | | | |
|------|--------|------|-------------|--|
| D41C | 2A16ED | LD | HL,(DEEP) | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えて指定するアドレスの内容を(WORK4+1)(WORK4)に格納する |
| D41F | EB | EX | DE,HL | |
| D420 | 2A32ED | LD | HL,(STACK, | |
| D423 | 19 | ADD | HL,DE | |
| D424 | D1 | POP | DE | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えさらに(WORK4+1)(WORK4)の内容を加えて指定するアドレスの内容を(WORK4+1)(WORK4)に格納する |
| D425 | 6E | LD | L,(HL) | |
| D426 | 2600 | LD | H,0 | |
| D428 | 221CED | LD | (WORK4),HL | |
| D42B | D5 | PUSH | DE | (BOARD+1)(BOARD)の内容に(WORK4+1)(WORK4)の内容を加えて指定するアドレスに(STONE1+1)(STONE1)の内容を格納する |
| D42C | 2A16ED | LD | HL,(DEEP) | |
| D42F | EB | EX | DE,HL | |
| D430 | 2A1CED | LD | HL,(WORK4) | |
| D433 | 19 | ADD | HL,DE | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えて指定するアドレスの内容を(WORK4+1)(WORK4)に格納する |
| D434 | EB | EX | DE,HL | |
| D435 | 2A32ED | LD | HL,(STACK) | |
| D438 | 19 | ADD | HL,DE | |
| D439 | D1 | POP | DE | (BOARD+1)(BOARD)の内容に(WORK4+1)(WORK4)の内容を加えて指定するアドレスに(STONE1+1)(STONE1)の内容を格納する |
| D43A | 6E | LD | L,(HL) | |
| D43B | 2600 | LD | H,0 | |
| D43D | 221CED | LD | (WORK4),HL | |
| D440 | 2A1CED | LD | HL,(WORK4) | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えて指定するアドレスの内容を(WORK4+1)(WORK4)に格納する |
| D443 | EB | EX | DE,HL | |
| D444 | 2A38ED | LD | HL,(BOARD) | |
| D447 | 19 | ADD | HL,DE | |
| D448 | E5 | PUSH | HL | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えて指定するアドレスの内容を(WORK4+1)(WORK4)に格納する |
| D449 | 2A0CED | LD | HL,(STONE1) | |
| D44C | EB | EX | DE,HL | |
| D44D | E1 | POP | HL | |
| D44E | 73 | LD | (HL),E | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えて指定するアドレスの内容を(WORK4+1)(WORK4)に格納する |
| D44F | D5 | PUSH | DE | |
| D450 | 2A16ED | LD | HL,(DEEP) | |
| D453 | EB | EX | DE,HL | |
| D454 | 2A32ED | LD | HL,(STACK) | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えさらに(WORK4+1)(WORK4)の内容を加えて指定するアドレスの内容を6で割った値を(COORD2+1)(COORD2)に格納する |
| D457 | 19 | ADD | HL,DE | |
| D458 | D1 | POP | DE | |
| D459 | 6E | LD | L,(HL) | |
| D45A | 2600 | LD | H,0 | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えて指定するアドレスの内容を(WORK4+1)(WORK4)に格納する |
| D45C | 221CED | LD | (WORK4),HL | |
| D45F | D5 | PUSH | DE | |
| D460 | 2A16ED | LD | HL,(DEEP) | |
| D463 | EB | EX | DE,HL | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えさらに(WORK4+1)(WORK4)の内容を加えて指定するアドレスの内容を6で割った値を(COORD2+1)(COORD2)に格納する |
| D464 | 2A1CED | LD | HL,(WORK4) | |
| D467 | 19 | ADD | HL,DE | |
| D468 | EB | EX | DE,HL | |
| D469 | 2A32ED | LD | HL,(STACK) | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えて指定するアドレスの内容を6で割った値を(COORD2+1)(COORD2)に格納する |
| D46C | 19 | ADD | HL,DE | |
| D46D | D1 | POP | DE | |
| D46E | 6E | LD | L,(HL) | |
| D46F | 2600 | LD | H,0 | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えて指定するアドレスの内容を6で割った値を(COORD2+1)(COORD2)に格納する |
| D471 | EB | EX | DE,HL | |
| D472 | 210600 | LD | HL,6 | |

| | | | | |
|------|--------|---------|-------------|---|
| D475 | CD24C1 | CALL | IDIV0 | |
| D478 | EB | EX | DE,HL | |
| D479 | 2212ED | LD | (COORD2),HL | |
| D47C | D5 | PUSH | DE | |
| D47D | D5 | PUSH | DE | |
| D47E | 2A16ED | LD | HL,(DEEP) | |
| D481 | EB | EX | DE,HL | |
| D482 | 2A1CED | LD | HL,(WORK4) | |
| D485 | 19 | ADD | HL,DE | |
| D486 | EB | EX | DE,HL | |
| D487 | 2A32ED | LD | HL,(STACK) | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えさらに(WORK4+1)(WORK4)の内容を加えて指定するアドレスの内容を6で割った剰余を(COORD1+1)(COORD1)に格納する |
| D48A | 19 | ADD | HL,DE | |
| D48B | D1 | POP | DE | |
| D48C | 6E | LD | L,(HL) | |
| D48D | 2600 | LD | H,0 | |
| D48F | EB | EX | DE,HL | |
| D490 | 210600 | LD | HL,6 | |
| D493 | CD24C1 | CALL | IDIV0 | |
| D496 | EB | EX | DE,HL | |
| D497 | D1 | POP | DE | |
| D498 | 2A4EED | LD | HL,(DIVWK) | |
| D49B | 2210ED | LD | (COORD1),HL | |
| D49E | CDCEC5 | CALL | VAL | 評価値を計算する |
| D4A1 | 2A16ED | LD | HL,(DEEP) | |
| D4A4 | EB | EX | DE,HL | |
| D4A5 | 210800 | LD | HL,8 | (DEEP+1)(DEEP)の内容に8を加える |
| D4A8 | 19 | ADD | HL,DE | |
| D4A9 | 2216ED | LD | (DEEP),HL | |
| D4AC | 210300 | LD | HL,3 | |
| D4AF | EB | EX | DE,HL | |
| D4B0 | 2A0CED | LD | HL,(STONE1) | (STONE1+1)(STONE)の内容が1の場合には2に2の場合には1に変更する |
| D4B3 | CD1EC1 | CALL | ISUB0 | |
| D4B6 | EB | EX | DE,HL | |
| D4B7 | 220CED | LD | (STONE1),HL | |
| D4BA | 210000 | LD | HL,0 | |
| D4BD | D1 | POP | DE | |
| D4BE | D5 | PUSH | DE | |
| D4BF | CD1EC1 | CALL | ISUB0 | |
| D4C2 | D1 | POP | DE | 『COMP4』にもどってくり返す |
| D4C3 | E1 | POP | HL | |
| D4C4 | FAC9D4 | JP | M,COMP11 | |
| D4C7 | E5 | PUSH | HL | |
| D4C8 | E9 | JP | (HL) | |
| | | | | |
| D4C9 | D5 | COMP11: | PUSH DE | |
| D4CA | 2A16ED | LD | HL,(DEEP) | |
| D4CD | EB | EX | DE,HL | |
| D4CE | 210100 | LD | HL,1 | (STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えさらに1を加えて指定するアドレスの内容を(WORK4+1)(WORK4)に格納する |
| D4D1 | 19 | ADD | HL,DE | |
| D4D2 | EB | EX | DE,HL | |
| D4D3 | 2A32ED | LD | HL,(STACK) | |

| | | | | |
|------|--------|---------|------|------------|
| D4D6 | 19 | | ADD | HL,DE |
| D4D7 | D1 | | POP | DE |
| D4D8 | 6E | | LD | L,(HL) |
| D4D9 | 2600 | | LD | H,0 |
| D4DB | 221CED | | LD | (WORK4),HL |
| D4DE | D5 | | PUSH | DE |
| D4DF | 2A1CED | | LD | HL,(WORK4) |
| D4E2 | EB | | EX | DE,HL |
| D4E3 | 210000 | | LD | HL,0 |
| D4E6 | CD1EC1 | | CALL | ISUB0 |
| D4E9 | B3 | | OR | E |
| D4EA | 110000 | | LD | DE,0 |
| D4ED | C2F1D4 | | JP | NZ,COMP12 |
| D4F0 | 1C | | INC | E |
| D4F1 | EB | COMP12: | EX | DE,HL |
| D4F2 | D1 | | POP | DE |
| D4F3 | EB | | EX | DE,HL |
| D4F4 | D5 | | PUSH | DE |
| D4F5 | 2A1CED | | LD | HL,(WORK4) |
| D4F8 | EB | | EX | DE,HL |
| D4F9 | 21FF00 | | LD | HL,0FFH |
| D4FC | CD1EC1 | | CALL | ISUB0 |
| D4FF | B3 | | OR | E |
| D500 | 110000 | | LD | DE,0 |
| D503 | C207D5 | | JP | NZ,COMP13 |
| D506 | 1C | | INC | E |
| D507 | EB | COMP13: | EX | DE,HL |
| D508 | D1 | | POP | DE |
| D509 | 19 | | ADD | HL,DE |
| D50A | EB | | EX | DE,HL |
| D50B | 7B | | LD | A,E |
| D50C | B2 | | OR | D |
| D50D | CA24D5 | | JP | Z,COMP14 |
| D510 | 2A16ED | | LD | HL,(DEEP) |
| D513 | EB | | EX | DE,HL |
| D514 | 210100 | | LD | HL,1 |
| D517 | 19 | | ADD | HL,DE |
| D518 | EB | | EX | DE,HL |
| D519 | 2A32ED | | LD | HL,(STACK) |
| D51C | 19 | | ADD | HL,DE |
| D51D | E5 | | PUSH | HL |
| D51E | 2A28ED | | LD | HL,(VALUE) |
| D521 | EB | | EX | DE,HL |
| D522 | E1 | | POP | HL |
| D523 | 73 | | LD | (HL),E |
| D524 | 210000 | COMP14: | LD | HL,0 |
| D527 | 2228ED | | LD | (VALUE),HL |
| D52A | 2A16ED | | LD | HL,(DEEP) |
| D52D | EB | | EX | DE,HL |
| D52E | 210000 | | LD | HL,0 |
| D531 | CD1EC1 | | CALL | ISUB0 |

(WORK4+1)(WORK4)の内容が0かまたはFFHの場合には(STACK+1)(STACK)の内容に(DEEP+1)(DEEP)の内容を加えさらに1を加えて指定するアドレスに(VALUE+1)(VALUE)の内容を格納する

(VALUE+1)(VALUE)に0を格納する


```

D534 B3          OR      E
D535 C25DD5      JP      NZ,COMP16
D538 210100      LD      HL,1
D53B D1          POP     DE
D53C D5          PUSH    DE
D53D CD1EC1      CALL    ISUB0
D540 D1          POP     DE
D541 E1          POP     HL
D542 FA47D5      JP      M,COMP15
D545 E5          PUSH    HL
D546 E9          JP      (HL)

;
D547 D5          COMP15:PUSH DE
D548 2A34ED      LD      HL,(COORD)
D54B EB          EX      DE,HL
D54C 210600      LD      HL,6
D54F CD24C1      CALL    IDIV0
D552 EB          EX      DE,HL
D553 D1          POP     DE
D554 2A4EED      LD      HL,(DIVWK)
D557 2234ED      LD      (COORD),HL
D55A C352D2      JP      MAIN58

;
D55D CD10C8      COMP16:CALL BACK
D560 D5          PUSH    DE
D561 2A16ED      LD      HL,(DEEP)
D564 EB          EX      DE,HL
D565 2A32ED      LD      HL,(STACK)
D568 19          ADD     HL,DE
D569 D1          POP     DE
D56A 6E          LD      L,(HL)
D56B 2600        LD      H,0
D56D 221CED      LD      (WORK4),HL
D570 D5          PUSH    DE
D571 2A1CED      LD      HL,(WORK4)
D574 EB          EX      DE,HL
D575 2A16ED      LD      HL,(DEEP)
D578 19          ADD     HL,DE
D579 EB          EX      DE,HL
D57A 2A32ED      LD      HL,(STACK)
D57D 19          ADD     HL,DE
D57E D1          POP     DE
D57F 6E          LD      L,(HL)
D580 2600        LD      H,0
D582 221CED      LD      (WORK4),HL
D585 2A1CED      LD      HL,(WORK4)
D588 EB          EX      DE,HL
D589 2A38ED      LD      HL,(BOARD)
D58C 19          ADD     HL,DE
D58D E5          PUSH    HL
D58E 210000      LD      HL,0
    
```

(DEEP+1)(DEEP)の内容が0
であれば『COMP4』からはじまるくり
返し処理を終了し(COORD+1)(C
OORD)の内容を6で割った剰余を同
じく(COORD+1)(COORD)に
格納後コンピュータの思考を中断する

} サブルーチン『BACK』を呼び出す

(STACK+1)(STACK)の内容
に(DEEP+1)(DEEP)の内容を
加えて指定するアドレスの内容を(WO
RK4+1)(WORK4)に格納する

(STACK+1)(STACK)の内容
に(WORK4+1)(WORK4)の内
容を加えさらに(DEEP+1)(DEE
P)の内容を加えて指定するアドレスの
内容を(WORK4+1)(WORK4)
に格納する

(BOARD+1)(BOARD)の内容
に(WORK4+1)(WORK4)の内
容を加えて指定するアドレスに0を格納
する

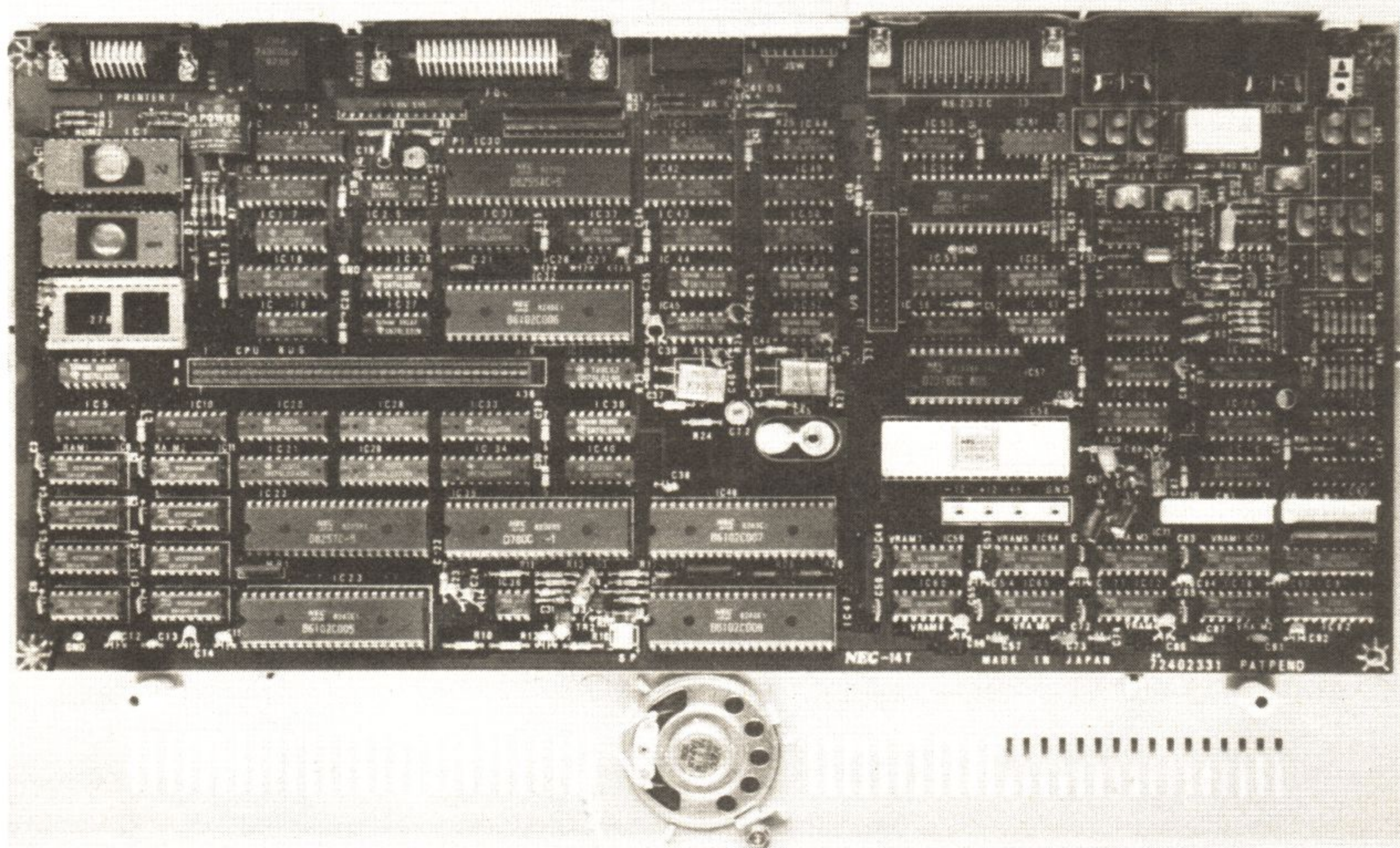

```

D591 EB          EX    DE,HL
D592 E1          POP   HL
D593 73          LD    (HL),E
D594 2A2CED       LD    HL,(ABFLAG)
D597 EB          EX    DE,HL
D598 210100       LD    HL,1
D59B CD1EC1       CALL  ISUB0
D59E B3          OR     E
D59F C2B1D5       JP    NZ,COMP17
D5A2 2A16ED       LD    HL,(DEEP)
D5A5 EB          EX    DE,HL
D5A6 2A32ED       LD    HL,(STACK)
D5A9 19          ADD   HL,DE
D5AA E5          PUSH  HL
D5AB 210200       LD    HL,2
D5AE EB          EX    DE,HL
D5AF E1          POP   HL
D5B0 73          LD    (HL),E
D5B1 C3E2D3       COMP17:JP  COMP9
                  ;
D5B4              END

```

(ABFLAG+1)(ABFLAG)の内容が1であれば(STACK+1)(STACK)の内容に(DEEP+1)(DEEP)を加えて指定するアドレスに2を格納する

『COMP9』にもどってくり返す



▲PC-8001mkIIの本体内部

CPU マイクロプロセッサの流れ

高性能機の時代

8086の誕生

Z80の普及から6809の発表に至るにつれて、8ビットの時代も隆盛を迎えました。

しかし、これら8ビットのマイクロプロセッサに対する要求が高まるにつれて、能力を超えるものが後をたちません。次に列記するものは、8ビットのマイクロプロセッサに対する不満のほんの一例です。ついこの間まで、メモリ容量が多いとか、実行速度が早いなどと言って喜んでいたのが嘘のようです。

- メモリ容量が少ない
- 処理速度が遅い
- 演算機能が乏しい(命令数の不足)
- アドレス指定の方法が少ない
- 他のCPUと協調する機能が欠けている
- 直接文字列を扱うための命令がない

そして、それらの不満を解消するため、1978年にインテル社より発表されたものが8086 (i8086=iAPX86)です。このような8086の生まれた背景には常に8ビット・マイクロプロセッサの欠点を解消するという意識が働いており、次に述べる二つの16ビット・マイクロプロセッサの生まれた背景とは若干異なります。

Z8000と68000

その後、ザイログ社、モトローラ社より、それぞれZ8000、68000 (MC68000) が発表されましたが、これらの背景となっているのは、自社の8ビット・マイクロプロセッサではなく、LSI-11です。LSI-11は、DEC社のミニ・コンピュータであるPDP-11の持つ機能を4チップのLSIにインプリメントしたものです。

したがって、Z8000および68000は、8ビット・マイクロプロセッサの拡張ではなくミニコンの1チップ化から生まれたもののなのです。

マイクロプロセッサとミニコン

実際に、8086、Z8000、68000などのいわゆる16ビット・マイクロプロセッサは、処理速度の点で、ほぼ1970年前半

の平均的なミニコンと同等以上の能力を持つにいたりました。扱うことのできるメモリの容量も、大半の16ビット・ミニコンを越えています。

8ビット・マイクロプロセッサが最も不得手としていた浮動小数点演算に関しても、インテル社の8087 (i8087) に代表される専用のコプロセッサや専用の周辺チップを用いることによって、浮動小数点演算の機能をハードウェアで備えているミニコンと並ぶ処理能力が得られるようになっていきます。

32ビットのマイクロプロセッサ

1981年、インテル社は3チップで構成されている32ビットのマイクロプロセッサ、iAPX432を発表しました。そして、このCPUは、2MIPSの処理速度を持っていると発表されています。おどろくべきことにこの処理速度は、現在の汎用大型コンピュータに匹敵する能力なのです。

さらに、iAPX432は仮想記憶機能を有しており、そのアドレス空間は約1兆バイトです。この性能は32ビットのスーパー・ミニコンを完全に追い越しています。

また、iAPX432の開発システムとして、インテル社からは、VAX-11上のADAコンパイラしかサポートされません。これは、iAPX432のアーキテクチャ自身がADA系コンパイラのオブジェクトを走らせることを意識しているためです。

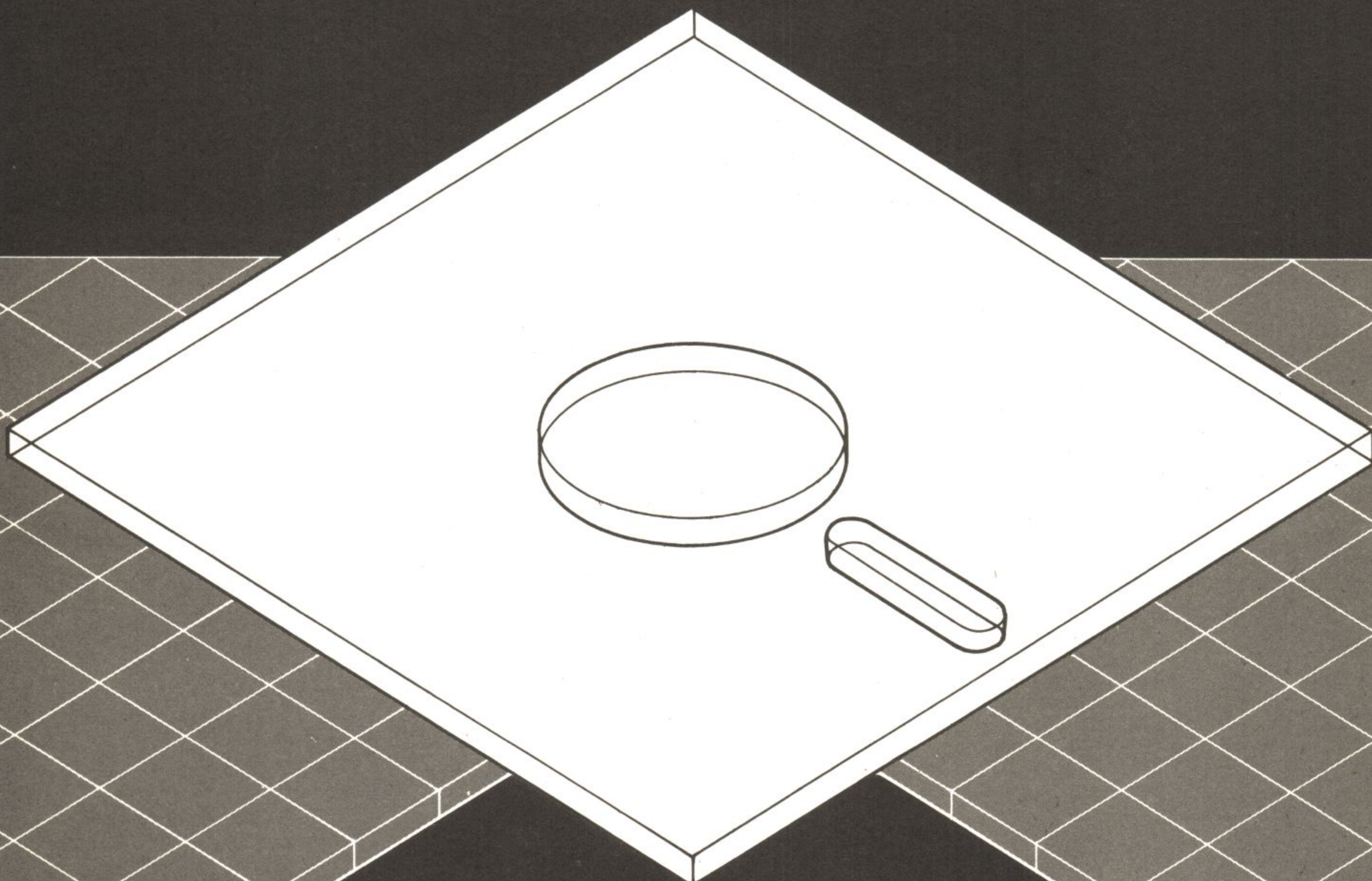
1982年3月2日、モトローラ社も16ビットの68000に続く32ビットのマイクロプロセッサとして、68010および68020を発表しました。

iAPX86(8086)とiAPX432の間には、影響を受けてはいるものの、同じインテル社の製品であること以外に特別な関係はありません。しかしモトローラ社の場合には上位互換性を意識した上で発展しており、68000を32ビットにしたものがまさに68020であると考えてよいでしょう。

このように、4004に始まったマイクロプロセッサは、より大きな処理能力とより高い機能を目指して流れてきました。これからも、より大きな処理能力とより高い機能を目指して流れて行くことでしょう。

5

第 ブロック



付 録

付録A μ COM-82インストラクション活用表

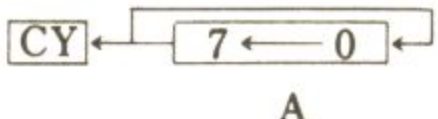
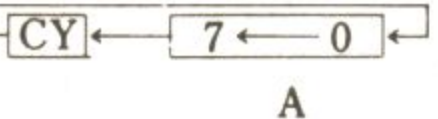
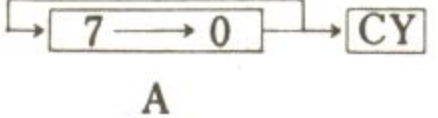
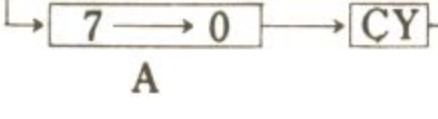
1. μ COM-82 インストラクション・セット

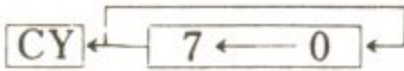
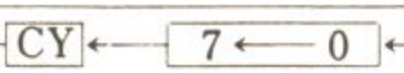
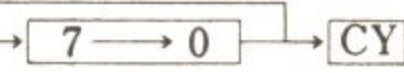
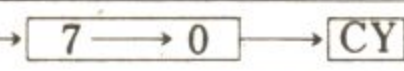
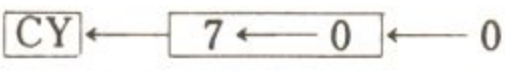
| 命令群 | ニーモニツク | オペレーション | フ ラ グ | | | | | バイト | ステート | OPコード | | | | |
|-------------------------|--|----------|-------|---|-----|-----|---|-----|------|-------|-----|-----|-----|---|
| | | | S | Z | H | P/V | N | | | C | 76 | 543 | 210 | |
| 8 ビット・ ロード 命令 | LD r, r' | r←r' | • | • | • | • | • | • | 1 | 4 | 01 | r | r' | Ⓔ |
| | LD r, n | r←n | • | • | • | • | • | • | 2 | 7 | 00 | r | 110 | Ⓔ |
| | | | | | | | | | | | ← | n | → | |
| | LD r, (HL) | r←(HL) | • | • | • | • | • | • | 1 | 7 | 01 | r | 110 | Ⓔ |
| | LD r, (IX+d) | r←(IX+d) | • | • | • | • | • | • | 3 | 19 | 11 | 011 | 101 | |
| | | | | | | | | | | | 01 | r | 110 | Ⓔ |
| | | | | | | | | | | | ← | d | → | |
| | LD r, (IY+d) | r←(IY+d) | • | • | • | • | • | • | 3 | 19 | 11 | 111 | 101 | |
| | | | | | | | | | | | 01 | r | 110 | Ⓔ |
| | | | | | | | | | | | ← | d | → | |
| | LD (HL), r | (HL)←r | • | • | • | • | • | • | 1 | 7 | 01 | 110 | r | Ⓔ |
| | LD (IX+d), r | (IX+d)←r | • | • | • | • | • | • | 3 | 19 | 11 | 011 | 101 | |
| | | | | | | | | | | | 01 | 110 | r | Ⓔ |
| | | | | | | | | | | | ← | d | → | |
| | LD (IY+d), r | (IY+d)←r | • | • | • | • | • | • | 3 | 19 | 11 | 111 | 101 | |
| | | | | | | | | | | | 01 | 110 | r | Ⓔ |
| | | | | | | | | | | | ← | d | → | |
| | LD (HL), n | (HL)←n | • | • | • | • | • | • | 2 | 10 | 00 | 110 | 110 | |
| | | | | | | | | | | | ← | n | → | |
| | LD (IX+d), n | (IX+d)←n | • | • | • | • | • | • | 4 | 19 | 11 | 011 | 101 | |
| | | | | | | | | | | | 00 | 110 | 110 | |
| | | | | | | | | | | | ← | d | → | |
| | | | | | | | | | | | ← | n | → | |
| | LD (IY+d), n | (IY+d)←n | • | • | • | • | • | • | 4 | 19 | 11 | 111 | 101 | |
| | | | | | | | | | | 00 | 110 | 110 | | |
| | | | | | | | | | | ← | d | → | | |
| | | | | | | | | | | ← | n | → | | |
| LD A, (BC) | A←(BC) | • | • | • | • | • | • | 1 | 7 | 00 | 001 | 010 | | |
| LD A, (DE) | A←(DE) | • | • | • | • | • | • | 1 | 7 | 00 | 011 | 010 | | |
| LD A, (nn) | A←(nn) | • | • | • | • | • | • | 3 | 13 | 00 | 111 | 010 | | |
| | | | | | | | | | | ← | n | → | | |
| | | | | | | | | | | ← | n | → | | |
| LD (BC), A | (BC)←A | • | • | • | • | • | • | 1 | 7 | 00 | 000 | 010 | | |
| LD (DE), A | (DE)←A | • | • | • | • | • | • | 1 | 7 | 00 | 010 | 010 | | |
| LD (nn), A | (nn)←A | • | • | • | • | • | • | 3 | 13 | 00 | 110 | 010 | | |
| | | | | | | | | | | ← | n | → | | |
| | | | | | | | | | | ← | n | → | | |
| LD A, I | A←I | ↑ | ↑ | 0 | IFF | 0 | • | 2 | 9 | 11 | 101 | 101 | | |
| | | | | | | | | | | 01 | 010 | 111 | | |
| LD A, R | A←R | ↑ | ↑ | 0 | IFF | 0 | • | 2 | 9 | 11 | 101 | 101 | | |
| | | | | | | | | | | 01 | 011 | 111 | | |
| LD I, A | I←A | • | • | • | • | • | • | 2 | 9 | 11 | 101 | 101 | | |
| | | | | | | | | | | 01 | 000 | 111 | | |
| LD R, A | R←A | • | • | • | • | • | • | 2 | 9 | 11 | 101 | 101 | | |
| | | | | | | | | | | 01 | 001 | 111 | | |
| 16 ビット・ ロード 命令 | LD dd, nn | dd←nn | • | • | • | • | • | • | 3 | 10 | 00 | dd0 | 001 | Ⓐ |
| | | | | | | | | | | | ← | n | → | |
| | | | | | | | | | | | ← | n | → | |
| | LD IX, nn | IX←nn | • | • | • | • | • | • | 4 | 14 | 11 | 011 | 101 | |
| | | | | | | | | | | | 00 | 100 | 001 | |
| | | | | | | | | | | | ← | n | → | |
| | | | | | | | | | | | ← | n | → | |
| | LD IY, nn | IY←nn | • | • | • | • | • | • | 4 | 14 | 11 | 111 | 101 | |
| | | | | | | | | | | | 00 | 100 | 001 | |
| | | | | | | | | | | | ← | n | → | |
| | | | | | | | | | | ← | n | → | | |
| LD HL, (nn) | H←(nn+1) L←(nn) | • | • | • | • | • | • | 3 | 16 | 00 | 101 | 010 | | |
| | | | | | | | | | | ← | n | → | | |
| | | | | | | | | | | ← | n | → | | |
| LD dd, (nn) | dd _H ←(nn+1) dd _L ←(nn) | • | • | • | • | • | • | 4 | 20 | 11 | 101 | 101 | | |
| | | | | | | | | | | 01 | dd1 | 011 | Ⓐ | |
| | | | | | | | | | | ← | n | → | | |
| | | | | | | | | | | ← | n | → | | |

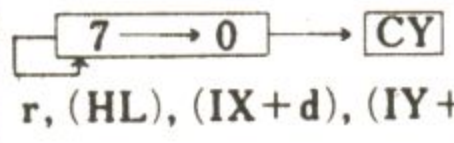
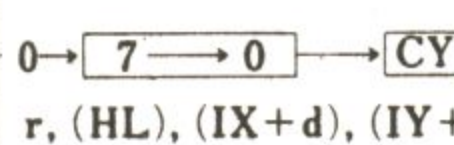
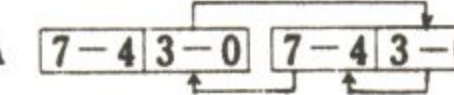
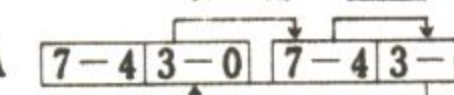
| 命令群 | ニーモニック | オペレーション | フ ラ グ | | | | | バイト | ステート | OPコード | | |
|---|-------------|---|-------|---|---|--------|---|-----|------|--------------------------|--|-----|
| | | | S | Z | H | P/V | N | C | | 76 | 543 | 210 |
| 16 ビット ・ ロ ー ド 命 令 | LD IX, (nn) | IX _H ←(nn+1) IX _L ←(nn) | • | • | • | • | • | • | 4 | 20 | 11 011 101 00 101 010 ← n → ← n → | |
| | LD IY, (nn) | IY _H ←(nn+1) IY _L ←(nn) | • | • | • | • | • | • | 4 | 20 | 11 111 101 00 101 010 ← n → ← n → | |
| | LD (nn), HL | (nn+1)←H (nn)←L | • | • | • | • | • | • | 3 | 16 | 00 100 010 ← n → ← n → | |
| | LD (nn), dd | (nn+1)←dd _H (nn)←dd _L | • | • | • | • | • | • | 4 | 20 | 11 101 101 01 dd0 011 ← n → ← n → | Ⓐ |
| | LD (nn), IX | (nn+1)←IX _H (nn)←IX _L | • | • | • | • | • | • | 4 | 20 | 11 011 101 00 100 010 ← n → ← n → | |
| | LD (nn), IY | (nn+1)←IY _H (nn)←IY _L | • | • | • | • | • | • | 4 | 20 | 11 111 101 00 100 010 ← n → ← n → | |
| | LD SP, HL | SP←HL | • | • | • | • | • | • | 1 | 6 | 11 111 001 | |
| | LD SP, IX | SP←IX | • | • | • | • | • | • | 2 | 10 | 11 011 101 11 111 001 | |
| | LD SP, IY | SP←IY | • | • | • | • | • | • | 2 | 10 | 11 111 101 11 111 001 | |
| | PUSH qq | (SP-2)←qq _L (SP-1)←qq _H | • | • | • | • | • | • | 1 | 11 | 11 qq0 101 | Ⓑ |
| | PUSH IX | (SP-2)←IX _L (SP-1)←IX _H | • | • | • | • | • | • | 2 | 15 | 11 011 101 11 100 101 | |
| | PUSH IY | (SP-2)←IY _L (SP-1)←IY _H | • | • | • | • | • | • | 2 | 15 | 11 111 101 11 100 101 | |
| | POP qq | qq _H ←(SP+1) qq _L ←(SP) | • | • | • | • | • | • | 1 | 10 | 11 qq0 001 | Ⓑ |
| | POP IX | IX _H ←(SP+1) IX _L ←(SP) | • | • | • | • | • | • | 2 | 14 | 11 011 101 11 100 001 | |
| | POP IY | IY _H ←(SP+1) IY _L ←(SP) | • | • | • | • | • | • | 2 | 14 | 11 111 101 11 100 001 | |
| エクス チ ェ ン ジ 命 令 | EX DE, HL | DE↔HL | • | • | • | • | • | • | 1 | 4 | 11 101 011 | |
| | EX AF, AF' | AF↔AF' | • | • | • | • | • | • | 1 | 4 | 00 001 000 | |
| | EXX | (BC↔BC') (DE↔DE') (HL↔HL') | • | • | • | • | • | • | 1 | 4 | 11 011 001 | |
| | EX (SP), HL | H↔(SP+1) L↔(SP) | • | • | • | • | • | • | 1 | 19 | 11 100 011 | |
| | EX (SP), IX | IX _H ↔(SP+1) IX _L ↔(SP) | • | • | • | • | • | • | 2 | 23 | 11 011 101 11 100 011 | |
| | EX (SP), IY | IY _H ↔(SP+1) IY _L ↔(SP) | • | • | • | • | • | • | 2 | 23 | 11 111 101 11 100 011 | |
| ブ ロ ッ ク 転 送 命 令 | LDI | (DE)←(HL) DE←DE+1 HL←HL+1 BC←BC-1 | • | • | 0 | ① ↓ | 0 | • | 2 | 16 | 11 101 101 10 100 000 | |
| | LDIR | (DE)←(HL) DE←DE+1 HL←HL+1 BC←BC-1 until BC=0 | • | • | 0 | 0 | 0 | • | 2 | 21 if BC≠0 16 if BC=0 | 11 101 101 10 110 000 | |

| 命令群 | ニーモニック | オペレーション | フ ラ グ | | | | | | バイト | ステート | OPコード | | | |
|--|----------------|---|-------------|-------------|-------------|-------------|---|---|-----|---|----------|------------|------------|---------------------|
| | | | S | Z | H | P/V | N | C | | | 76 | 543 | 210 | |
| ブロック転送命令 | LDD | (DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1 | • | • | 0 | ① ↓ | 0 | • | 2 | 16 | 11 10 | 101 101 | 101 000 | |
| | LDDR | (DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1 until BC=0 | • | • | 0 | 0 | 0 | • | 2 | 21 if BC≠0 16 if BC=0 | 11 10 | 101 111 | 101 000 | |
| ブロック・サーチ命令 | CPI | A←(HL) HL←HL+1 BC←BC-1 | ↑ ② ↓ | ↑ ② ↓ | ↑ ② ↓ | ↑ ① ↓ | 1 | • | 2 | 16 | 11 10 | 101 100 | 101 001 | |
| | CPIR | A←(HL) HL←HL+1 BC←BC-1 until A=(HL) or BC=0 | ↑ ② ↓ | ↑ ② ↓ | ↑ ② ↓ | ↑ ① ↓ | 1 | • | 2 | 21 if BC≠0 and A≠(HL) 16 if BC=0 or A=(HL) | 11 10 | 101 110 | 101 001 | |
| | CPD | A←(HL) HL←HL-1 BC←BC-1 | ↑ ② ↓ | ↑ ② ↓ | ↑ ② ↓ | ↑ ① ↓ | 1 | • | 2 | 16 | 11 10 | 101 101 | 101 001 | |
| | CPDR | A←(HL) HL←HL-1 BC←BC-1 until A=(HL) or BC=0 | ↑ ② ↓ | ↑ ② ↓ | ↑ ② ↓ | ↑ ① ↓ | 1 | • | 2 | 21 if BC≠0 and A≠(HL) 16 if BC=0 or A=(HL) | 11 10 | 101 111 | 101 001 | |
| 8 ビット 算 術 論 理 演 算 命 令 | ADD A, r | A←A+r | ↑ | ↑ | ↑ | ↑ | V | 0 | ↑ | 1 | 4 | 10 | 000 | r ⑤ |
| | ADD A, n | A←A+n | ↑ | ↑ | ↑ | ↑ | V | 0 | ↑ | 2 | 7 | 11 | 000 | 110 ← n → |
| | ADD A, (HL) | A←A+(HL) | ↑ | ↑ | ↑ | ↑ | V | 0 | ↑ | 1 | 7 | 10 | 000 | 110 |
| | ADD A, (IX+d) | A←A+(IX+d) | ↑ | ↑ | ↑ | ↑ | V | 0 | ↑ | 3 | 19 | 11 10 | 011 000 | 101 110 ← d → |
| | ADD, A, (IY+d) | A←A+(IY+d) | ↑ | ↑ | ↑ | ↑ | V | 0 | ↑ | 3 | 19 | 11 10 | 111 000 | 101 110 ← d → |
| | ADC A, r | A←A+r+CY | ↑ | ↑ | ↑ | ↑ | V | 0 | ↑ | 1 | 4 | 10 | 001 | r ⑤ |
| | ADC A, n | A←A+n+CY | ↑ | ↑ | ↑ | ↑ | V | 0 | ↑ | 2 | 7 | 11 | 001 | 110 ← n → |
| | ADC A, (HL) | A←A+(HL)+CY | ↑ | ↑ | ↑ | ↑ | V | 0 | ↑ | 1 | 7 | 10 | 001 | 110 |
| | ADC A, (IX+d) | A←A+(IX+d)+CY | ↑ | ↑ | ↑ | ↑ | V | 0 | ↑ | 3 | 19 | 11 10 | 011 001 | 101 110 ← d → |
| | ADC A, (IY+d) | A←A+(IY+d)+CY | ↑ | ↑ | ↑ | ↑ | V | 0 | ↑ | 3 | 19 | 11 10 | 111 001 | 101 110 ← d → |
| | SUB r | A←A-r | ↑ | ↑ | ↑ | ↑ | V | 1 | ↑ | 1 | 4 | 10 | 010 | r ⑤ |
| | SUB n | A←A-n | ↑ | ↑ | ↑ | ↑ | V | 1 | ↑ | 2 | 7 | 11 | 010 | 110 ← n → |
| | SUB (HL) | A←A-(HL) | ↑ | ↑ | ↑ | ↑ | V | 1 | ↑ | 1 | 7 | 10 | 010 | 110 |
| | SUB (IX+d) | A←A-(IX+d) | ↑ | ↑ | ↑ | ↑ | V | 1 | ↑ | 3 | 19 | 11 10 | 011 010 | 101 110 ← d → |
| | SUB (IY+d) | A←A-(IY+d) | ↑ | ↑ | ↑ | ↑ | V | 1 | ↑ | 3 | 19 | 11 10 | 111 010 | 101 110 ← d → |
| | SBC A, r | A←A-r-CY | ↑ | ↑ | ↑ | ↑ | V | 1 | ↑ | 1 | 4 | 10 | 011 | r ⑤ |
| | SBC A, n | A←A-n-CY | ↑ | ↑ | ↑ | ↑ | V | 1 | ↑ | 2 | 7 | 11 | 011 | 110 ← n → |
| | SBC A, (HL) | A←A-(HL)-CY | ↑ | ↑ | ↑ | ↑ | V | 1 | ↑ | 1 | 7 | 10 | 011 | 110 |

| 命令群 | ニーモニック | オペレーション | フ ラ グ | | | | | | バイト | ステート | OPコード | | |
|--|---------------|--------------------------------|-------|---|---|-----|---|---|-----|------|-------|-----|-------|
| | | | S | Z | H | P/V | N | C | | | 76 | 543 | 210 |
| 8 ビ ッ ト 算 術 論 理 演 算 命 令 | SBC A, (IX+d) | $A \leftarrow A - (IX+d) - CY$ | ↑ | ↑ | ↑ | V | 1 | ↑ | 3 | 19 | 11 | 011 | 101 |
| | | | | | | | | | | | 10 | 011 | 110 |
| | | | | | | | | | | | ← | d | → |
| | SBC A, (IY+d) | $A \leftarrow A - (IY+d) - CY$ | ↑ | ↑ | ↑ | V | 1 | ↑ | 3 | 19 | 11 | 111 | 101 |
| | | | | | | | | | | | 10 | 011 | 110 |
| | | | | | | | | | | | ← | d | → |
| | AND r | $A \leftarrow A \wedge r$ | ↑ | ↑ | 1 | P | 0 | 0 | 1 | 4 | 10 | 100 | r ⑤ |
| | AND n | $A \leftarrow A \wedge n$ | ↑ | ↑ | 1 | P | 0 | 0 | 2 | 7 | 11 | 100 | 110 |
| | | | | | | | | | | | ← | n | → |
| | AND (HL) | $A \leftarrow A \wedge (HL)$ | ↑ | ↑ | 1 | P | 0 | 0 | 1 | 7 | 10 | 100 | 110 |
| | AND (IX+d) | $A \leftarrow A \wedge (IX+d)$ | ↑ | ↑ | 1 | P | 0 | 0 | 3 | 19 | 11 | 011 | 101 |
| | | | | | | | | | | | 10 | 100 | 110 |
| | | | | | | | | | | | ← | d | → |
| | AND (IY+d) | $A \leftarrow A \wedge (IY+d)$ | ↑ | ↑ | 1 | P | 0 | 0 | 3 | 19 | 11 | 111 | 101 |
| | | | | | | | | | | | 10 | 100 | 110 |
| | | | | | | | | | | | ← | d | → |
| | OR r | $A \leftarrow A \vee r$ | ↑ | ↑ | 0 | P | 0 | 0 | 1 | 4 | 10 | 110 | r ⑤ |
| | OR n | $A \leftarrow A \vee n$ | ↑ | ↑ | 0 | P | 0 | 0 | 2 | 7 | 11 | 110 | 110 |
| | | | | | | | | | | | ← | n | → |
| | OR (HL) | $A \leftarrow A \vee (HL)$ | ↑ | ↑ | 0 | P | 0 | 0 | 1 | 7 | 10 | 110 | 110 |
| | OR (IX+d) | $A \leftarrow A \vee (IX+d)$ | ↑ | ↑ | 0 | P | 0 | 0 | 3 | 19 | 11 | 011 | 101 |
| | | | | | | | | | | | 10 | 110 | 110 |
| | | | | | | | | | | | ← | d | → |
| | OR (IY+d) | $A \leftarrow A \vee (IY+d)$ | ↑ | ↑ | 0 | P | 0 | 0 | 3 | 19 | 11 | 111 | 101 |
| | | | | | | | | | | | 10 | 110 | 110 |
| | | | | | | | | | | | ← | d | → |
| | XOR r | $A \leftarrow A \oplus r$ | ↑ | ↑ | 0 | P | 0 | 0 | 1 | 4 | 10 | 101 | r ⑤ |
| | XOR n | $A \leftarrow A \oplus n$ | ↑ | ↑ | 0 | P | 0 | 0 | 2 | 7 | 11 | 101 | 110 |
| | | | | | | | | | | | ← | n | → |
| | XOR (HL) | $A \leftarrow A \oplus (HL)$ | ↑ | ↑ | 0 | P | 0 | 0 | 1 | 7 | 10 | 101 | 110 |
| | XOR (IX+d) | $A \leftarrow A \oplus (IX+d)$ | ↑ | ↑ | 0 | P | 0 | 0 | 3 | 19 | 11 | 011 | 101 |
| | | | | | | | | | | | 10 | 101 | 110 |
| | | | | | | | | | | | ← | d | → |
| | XOR (IY+d) | $A \leftarrow A \oplus (IY+d)$ | ↑ | ↑ | 0 | P | 0 | 0 | 3 | 19 | 11 | 111 | 101 |
| | | | | | | | | | | | 10 | 101 | 110 |
| | | | | | | | | | | | ← | d | → |
| | CP r | $A - r$ | ↑ | ↑ | ↑ | V | 1 | ↑ | 1 | 4 | 10 | 111 | r ⑤ |
| | CP n | $A - n$ | ↑ | ↑ | ↑ | V | 1 | ↑ | 2 | 7 | 11 | 111 | 110 |
| | | | | | | | | | | | ← | n | → |
| | CP (HL) | $A - (HL)$ | ↑ | ↑ | ↑ | V | 1 | ↑ | 1 | 7 | 10 | 111 | 110 |
| | CP (IX+d) | $A - (IX+d)$ | ↑ | ↑ | ↑ | V | 1 | ↑ | 3 | 19 | 11 | 011 | 101 |
| | | | | | | | | | | | 10 | 111 | 110 |
| | | | | | | | | | | | ← | d | → |
| | CP (IY+d) | $A - (IY+d)$ | ↑ | ↑ | ↑ | V | 1 | ↑ | 3 | 19 | 11 | 111 | 101 |
| | | | | | | | | | | | 10 | 111 | 110 |
| | | | | | | | | | | | ← | d | → |
| | INC r | $r \leftarrow r + 1$ | ↑ | ↑ | ↑ | V | 0 | • | 1 | 4 | 00 | r | 100 ⑤ |
| | INC (HL) | $(HL) \leftarrow (HL) + 1$ | ↑ | ↑ | ↑ | V | 0 | • | 1 | 11 | 00 | 110 | 100 |
| | INC (IX+d) | $(IX+d) \leftarrow (IX+d) + 1$ | ↑ | ↑ | ↑ | V | 0 | • | 3 | 23 | 11 | 011 | 101 |
| | | | | | | | | | | | 00 | 110 | 100 |
| | | | | | | | | | | | ← | d | → |
| | INC (IY+d) | $(IY+d) \leftarrow (IX+d) + 1$ | ↑ | ↑ | ↑ | V | 0 | • | 3 | 23 | 11 | 111 | 101 |
| | | | | | | | | | | | 00 | 110 | 100 |
| | | | | | | | | | | | ← | d | → |
| | DEC r | $r \leftarrow r - 1$ | ↑ | ↑ | ↑ | V | 1 | • | 1 | 4 | 00 | r | 101 ⑤ |
| | DEC (HL) | $(HL) \leftarrow (HL) - 1$ | ↑ | ↑ | ↑ | V | 1 | • | 1 | 11 | 00 | 110 | 101 |
| | DEC (IX+d) | $(IX+d) \leftarrow (IX+d) - 1$ | ↑ | ↑ | ↑ | V | 1 | • | 3 | 23 | 11 | 011 | 101 |
| | | | | | | | | | | | 00 | 110 | 101 |
| | | | | | | | | | | | ← | d | → |

| 命令群 | ニーモニック | オペレーション | フ ラ グ | | | | | | バイト | ステート | OPコード | | |
|---|------------|---|-------|---|---|-----|---|---|-----|------|-------|-----|-----------|
| | | | S | Z | H | P/V | N | C | | | 76 | 543 | 210 |
| 演算8 算術ビ 命論ッ 合理ト | DEC (IY+d) | $(IY+d) \leftarrow (IY+d) - 1$ | ↑ | ↑ | ↑ | V | 1 | • | 3 | 23 | 11 | 111 | 101 |
| | | | | | | | | | | | 00 | 110 | 101 |
| 16 ビ ッ ト 算 術 演 算 命 令 | ADD HL, ss | $HL \leftarrow HL + ss$ | • | • | × | • | 0 | ↑ | 1 | 11 | 00 | ss1 | 001 - (A) |
| | ADC HL, ss | $HL \leftarrow HL + ss + CY$ | ↑ | ↑ | × | V | 0 | ↑ | 2 | 15 | 11 | 101 | 101 |
| | SBC HL, ss | $HL \leftarrow HL - ss - CY$ | ↑ | ↑ | × | V | 1 | ↑ | 2 | 15 | 01 | ss1 | 010 (A) |
| | ADD IX, pp | $IX \leftarrow IX + pp$ | • | • | × | • | 0 | ↑ | 2 | 15 | 11 | 101 | 101 |
| | ADD IY, rr | $IY \leftarrow IY + rr$ | • | • | × | • | 0 | ↑ | 2 | 15 | 01 | ss0 | 010 (A) |
| | INC ss | $ss \leftarrow ss + 1$ | • | • | • | • | • | • | 1 | 6 | 11 | 011 | 101 |
| | INC IX | $IX \leftarrow IX + 1$ | • | • | • | • | • | • | 2 | 10 | 00 | pp1 | 001 (C) |
| | INC IY | $IY \leftarrow IY + 1$ | • | • | • | • | • | • | 2 | 10 | 11 | 101 | 101 |
| | DEC ss | $ss \leftarrow ss - 1$ | • | • | • | • | • | • | 1 | 6 | 00 | rr1 | 001 (D) |
| | DEC IX | $IX \leftarrow IX - 1$ | • | • | • | • | • | • | 2 | 10 | 00 | ss0 | 011 (A) |
| | DEC IY | $IY \leftarrow IY - 1$ | • | • | • | • | • | • | 2 | 10 | 11 | 011 | 101 |
| | | | | | | | | | | | 00 | 100 | 011 |
| | | | | | | | | | | | 11 | 011 | 101 |
| ア キ ュ ム レ ー タ 操 作 命 令 | DAA | Decimal adjust Acc | ↑ | ↑ | ↑ | P | • | ↑ | 1 | 4 | 00 | ss1 | 011 (A) |
| | CPL | $A \leftarrow \bar{A}$ | • | • | 1 | • | 1 | • | 1 | 4 | 11 | 101 | 101 |
| | NEG | $A \leftarrow \bar{A} + 1$ | ↑ | ↑ | ↑ | V | 1 | ↑ | 2 | 8 | 01 | 000 | 100 |
| | CCF | $CY \leftarrow \bar{CY}$ | • | • | × | • | 0 | ↑ | 1 | 4 | 00 | 111 | 111 |
| | SCF | $CY \leftarrow 1$ | • | • | 0 | • | 0 | 1 | 1 | 4 | 00 | 110 | 111 |
| C P U コ ン ト ロ ー ル 命 令 | NOP | No operation | • | • | • | • | • | • | 1 | 4 | 00 | 000 | 000 |
| | HALT | CPU halted | • | • | • | • | • | • | 1 | 4 | 01 | 110 | 110 |
| | DI | $IFF \leftarrow 0$ | • | • | • | • | • | • | 1 | 4 | 11 | 110 | 011 |
| | EI | $IFF \leftarrow 1$ | • | • | • | • | • | • | 1 | 4 | 11 | 111 | 011 |
| | IM 0 | Set interrupt mode 0 | • | • | • | • | • | • | 2 | 8 | 11 | 101 | 101 |
| | IM 1 | Set interrupt mode 1 | • | • | • | • | • | • | 2 | 8 | 01 | 000 | 110 |
| | IM 2 | Set interrupt mode 2 | • | • | • | • | • | • | 2 | 8 | 11 | 101 | 101 |
| ロ ー テ ー ト ・ シ フ ト 命 令 | RLCA |  | • | • | 0 | • | 0 | ↑ | 1 | 4 | 01 | 011 | 110 |
| | RLA |  | • | • | 0 | • | 0 | ↑ | 1 | 4 | 00 | 000 | 111 |
| | RRCA |  | • | • | 0 | • | 0 | ↑ | 1 | 4 | 00 | 010 | 111 |
| | RRA |  | • | • | 0 | • | 0 | ↑ | 1 | 4 | 00 | 001 | 111 |

| 命令群 | ニーモニック | オペレーション | フ ラ グ | | | | | | バイト | ステート | OPコード | | |
|---|------------|--|-------|---|---|-----|---|---|-----|------|------------|-----|-----|
| | | | S | Z | H | P/V | N | C | | | 76 | 543 | 210 |
| ロー レ ジ ス タ ー ・ シ フ ト 命 令 | RLC r |  r, (HL), (IX+d), (IY+d) | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 8 | 11 001 011 | Ⓔ | r |
| | RLC (HL) | | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 15 | 11 001 011 | | |
| | RLC (IX+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 011 101 | | |
| | RLC (IY+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 001 011 | | |
| | RL r |  r, (HL), (IX+d), (IY+d) | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 8 | 11 001 011 | Ⓔ | r |
| | RL (HL) | | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 15 | 11 001 011 | | |
| | RL (IX+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 011 101 | | |
| | RL (IY+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 001 011 | | |
| | RRC r |  r, (HL), (IX+d), (IY+d) | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 8 | 11 001 011 | Ⓔ | r |
| | RRC (HL) | | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 15 | 11 001 011 | | |
| | RRC (IX+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 011 101 | | |
| | RRC (IY+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 001 011 | | |
| | RR r |  r, (HL), (IX+d), (IY+d) | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 8 | 11 001 011 | Ⓔ | r |
| | RR (HL) | | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 15 | 11 001 011 | | |
| | RR (IX+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 011 101 | | |
| | RR (IY+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 001 011 | | |
| | SLA r |  r, (HL), (IX+d), (IY+d) | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 8 | 11 001 011 | Ⓔ | r |
| | SLA (HL) | | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 15 | 11 001 011 | | |
| | SLA (IX+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 011 101 | | |
| | SLA (IY+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 001 011 | | |

| 命令群 | ニーモニック | オペレーション | フ ラ グ | | | | | | バイト | ステート | OPコード | | |
|-------------------------|---|---|-------|---|-------|---|---|----|-----|------------|------------|------------|-----|
| | | | S | Z | H P/V | N | C | 76 | | | 543 | 210 | |
| ロー シフト 命令 | SRA r |  | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 8 | 11 001 011 | 00 101 r | Ⓔ |
| | SRA (HL) | | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 15 | 11 001 011 | 00 101 110 | |
| | SRA (IX+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 011 101 | 11 001 011 | |
| | SRA (IY+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 111 101 | 11 001 011 | |
| | SRL r |  | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 8 | 11 001 011 | 00 111 r | Ⓔ |
| | SRL (HL) | | ↑ | ↑ | 0 | P | 0 | ↑ | 2 | 15 | 11 001 011 | 00 111 110 | |
| | SRL (IX+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 011 101 | 11 001 011 | |
| | SRL (IY+d) | | ↑ | ↑ | 0 | P | 0 | ↑ | 4 | 23 | 11 111 101 | 11 001 011 | |
| | RLD | A  | ↑ | ↑ | 0 | P | 0 | • | 2 | 18 | 11 101 101 | 01 101 111 | |
| RRD | A  | ↑ | ↑ | 0 | P | 0 | • | 2 | 18 | 11 101 101 | 01 100 111 | | |
| ビット 操 作 命 令 | BIT b, r | $Z \leftarrow \overline{r_b}$ | × | ↑ | 1 | × | 0 | • | 2 | 8 | 11 001 011 | 01 b r | Ⓔ Ⓕ |
| | BIT b, (HL) | $Z \leftarrow \overline{(HL)_b}$ | × | ↑ | 1 | × | 0 | • | 2 | 12 | 11 001 011 | 01 b 110 | Ⓕ |
| | BIT b, (IX+d) | $Z \leftarrow \overline{(IX+d)_b}$ | × | ↑ | 1 | × | 0 | • | 4 | 20 | 11 011 101 | 11 001 011 | |
| | BIT b, (IY+d) | $Z \leftarrow \overline{(IY+d)_b}$ | × | ↑ | 1 | × | 0 | • | 4 | 20 | 11 111 101 | 11 001 011 | |
| | SET b, r | $r_b \leftarrow 1$ | • | • | • | • | • | • | 2 | 8 | 11 001 011 | 11 b r | Ⓔ Ⓕ |
| | SET b, (HL) | $(HL)_b \leftarrow 1$ | • | • | • | • | • | • | 2 | 15 | 11 001 011 | 11 b 110 | Ⓕ |
| | SET b, (IX+d) | $(IX+d)_b \leftarrow 1$ | • | • | • | • | • | • | 4 | 23 | 11 011 101 | 11 001 011 | |
| | SET b, (IY+d) | $(IY+d)_b \leftarrow 1$ | • | • | • | • | • | • | 4 | 23 | 11 111 101 | 11 001 011 | |
| | RES b, r | $r_b \leftarrow 0$ | • | • | • | • | • | • | 2 | 8 | 11 001 011 | 10 b r | Ⓔ Ⓕ |
| | RES b, (HL) | $(HL)_b \leftarrow 0$ | • | • | • | • | • | • | 2 | 15 | 11 001 011 | 10 b 110 | Ⓕ |
| | RES b, (IX+d) | $(IX+d)_b \leftarrow 0$ | • | • | • | • | • | • | 4 | 23 | 11 011 101 | 11 001 011 | |
| | RES b, (IY+d) | $(IY+d)_b \leftarrow 0$ | • | • | • | • | • | • | 4 | 23 | 11 111 101 | 11 001 011 | |

| 命令群 | ニーモニック | オペレーション | フ ラ グ | | | | | バイト | ステート | O P コード | | |
|-----------------|-------------|--|-------|---|---|------|---|-----|---------------------------------------|-------------|------------|-----|
| | | | S | Z | H | P/VN | C | | | 76 | 543 | 210 |
| ジャンプ・コール・リターン命令 | JP nn | PC←nn | . | . | . | . | . | 3 | 10 | 11 000 011 | ← n → | |
| | JP cc, nn | If cc is true PC←nn Otherwise continue | . | . | . | . | . | 3 | 10 | 11 cc 010 ⑥ | ← n → | |
| | JR e | PC←PC+e | . | . | . | . | . | 2 | 12 | 00 011 000 | ← e-2 → | |
| | JR C, e | If C=0 continue If C=1 PC←PC+e | . | . | . | . | . | 2 | 7 if C=0 12 if C=1 | 00 111 000 | ← e-2 → | |
| | JR NC, e | If C=1 continue If C=0 PC←PC+e | . | . | . | . | . | 2 | 7 if C=1 12 if C=0 | 00 110 000 | ← e-2 → | |
| | JR Z, e | If Z=0 continue If Z=1 PC←PC+e | . | . | . | . | . | 2 | 7 if Z=0 12 if Z=1 | 00 101 000 | ← e-2 → | |
| | JR NZ, e | If Z=1 continue If Z=0 PC←PC+e | . | . | . | . | . | 2 | 7 if Z=1 12 if Z=0 | 00 100 000 | ← e-2 → | |
| | JP (HL) | PC←HL | . | . | . | . | . | 1 | 4 | 11 101 001 | | |
| | JP (IX) | PC←IX | . | . | . | . | . | 2 | 8 | 11 011 101 | | |
| | JP (IY) | PC←IY | . | . | . | . | . | 2 | 8 | 11 111 101 | | |
| | DJNZ e | B←B-1 if B=0 continue if B≠0 PC←PC+e | . | . | . | . | . | 2 | 8 if B=0 13 if B≠0 | 00 010 000 | ← e-2 → | |
| | CALL nn | (SP-1)←PC _H (SP-2)←PC _L PC←nn | . | . | . | . | . | 3 | 17 | 11 001 101 | ← n → | |
| | CALL cc, nn | If cc is false continue otherwise same as CALL nn | . | . | . | . | . | 3 | 10 if cc is false 17 if cc is true | 11 cc 100 ⑥ | ← n → | |
| | RET | PC _L ←(SP) PC _H ←(SP+1) | . | . | . | . | . | 1 | 10 | 11 001 001 | | |
| | RET cc | If cc is false continue otherwise same as RET | . | . | . | . | . | 1 | 5 if cc is false 11 if cc is true | 11 cc 000 ⑥ | | |
| | RETI | Return from interrupt | . | . | . | . | . | 2 | 14 | 11 101 101 | | |
| | RETN | Return from non maskable interrupt | . | . | . | . | . | 2 | 14 | 01 001 101 | | |
| | RST p | (SP-1)←PC _H (SP-2)←PC _L PC _H ←0 PC _L ←p | . | . | . | . | . | 1 | 11 | 11 101 101 | 01 000 101 | |
| 入出力命令 | IN A, n | A←(n) A ₀₋₇ ←n A ₈₋₁₅ ←A | . | . | . | . | . | 2 | 11 | 11 011 011 | ← n → | |
| | IN r, (C) | r←(C) if r=110 only the flags will be affected A ₀₋₇ ←C A ₈₋₁₅ ←B | ↑ | ↑ | ↑ | P | 0 | 2 | 12 | 11 101 101 | 01 r 000 ⑥ | |
| | INI | (HL)←(C) B←B-1 HL←HL+1 A ₀₋₇ ←C A ₈₋₁₅ ←B | × | ③ | × | × | 1 | 2 | 16 | 11 101 101 | 10 100 010 | |
| | INIR | (HL)←(C) B←B-1 HL←HL+1 until B=0 A ₀₋₇ ←C A ₈₋₁₅ ←B | × | 1 | × | × | 1 | 2 | 21 if B≠0 16 if B=0 | 11 101 101 | 10 110 010 | |
| | IND | (HL)←(C) B←B-1 HL←HL-1 A ₀₋₇ ←C A ₈₋₁₅ ←B | × | ③ | × | × | 1 | 2 | 16 | 11 101 101 | 10 101 010 | |

| 命令群 | ニーモニック | オペレーション | フ ラ グ | | | | | | バイト | ステート | OPコード | | |
|-------|------------|---|-------|--------|---|-----|---|---|-----|----------------------------|-------------------------------------|-----|-----|
| | | | S | Z | H | P/V | N | C | | | 76 | 543 | 210 |
| 入出力命令 | INDR | (HL) ← (C) B ← B - 1 HL ← HL - 1 until B = 0 A ₀₋₇ ← C A ₈₋₁₅ ← B | × | 1 | × | × | 1 | • | 2 | 21 if B ≠ 0 16 if B = 0 | 11 101 101 10 111 010 | | |
| | OUT n, A | (n) ← A A ₀₋₇ ← n A ₈₋₁₅ ← A | • | • | • | • | • | • | 2 | 11 | 11 010 011 ← n → | | |
| | OUT (C), r | (C) ← r A ₀₋₇ ← C A ₈₋₁₅ ← B | • | • | • | • | • | • | 2 | 12 | 11 101 101 01 r 001 ^⑤ | | |
| | OUTI | (C) ← (HL) B ← B - 1 HL ← HL + 1 A ₀₋₇ ← C A ₈₋₁₅ ← B | × | ③ ↓ | × | × | 1 | • | 2 | 16 | 11 101 101 10 100 011 | | |
| | OTIR | (C) ← (HL) B ← B - 1 HL ← HL + 1 until B = 0 A ₀₋₇ ← C A ₈₋₁₅ ← B | × | 1 | × | × | 1 | • | 2 | 21 if B ≠ 0 16 if B = 0 | 11 101 101 10 110 011 | | |
| | OUTD | (C) ← (HL) B ← B - 1 HL ← HL - 1 A ₀₋₇ ← C A ₈₋₁₅ ← B | × | ③ ↓ | × | × | 1 | • | 2 | 16 | 11 101 101 10 101 011 | | |
| | OTDR | (C) ← (HL) B ← B - 1 HL ← HL - 1 until B = 0 A ₀₋₇ ← C A ₈₋₁₅ ← B | × | 1 | × | × | 1 | • | 2 | 21 if B ≠ 0 16 if B = 0 | 11 101 101 10 111 011 | | |

| ① | | ② | | ③ | | ④ | | ⑤ | | ⑥ | | ⑦ | | | ⑧ | |
|--|----------|-----|----|-----|----|-----|----|-----|-------|-----|-----|-----|-----------------|------|-----|-----|
| Reg | ss dd | Reg | qq | Reg | pp | Reg | rr | Reg | r, r' | Bit | b | cc | Condition | Flag | P | t |
| BC | 00 | BC | 00 | BC | 00 | BC | 00 | B | 000 | 0 | 000 | 000 | NZ Non Zero | Z | 00H | 000 |
| DE | 01 | DE | 01 | DE | 01 | DE | 01 | C | 001 | 1 | 001 | 001 | Z Zero | Z | 08H | 001 |
| HL | 10 | HL | 10 | IX | 10 | IX | 10 | D | 010 | 2 | 010 | 010 | NC Non Carry | C | 10H | 010 |
| SP | 11 | AF | 11 | SP | 11 | SP | 11 | E | 011 | 3 | 011 | 011 | C Carry | C | 18H | 011 |
| d, n : 8ビット・イミディエト・データ e : 相対アドレッシングの変位置 e-2 : eの実効変位置 | | | | | | | | H | 100 | 4 | 100 | 100 | PO Parity Odd | P/V | 20H | 100 |
| | | | | | | | | L | 101 | 5 | 101 | 101 | PE Parity Even | P/V | 28H | 101 |
| | | | | | | | | A | 111 | 6 | 110 | 110 | P Sign Positive | S | 30H | 110 |
| | | | | | | | | | | 7 | 111 | 111 | M Sign Negative | S | 38H | 111 |

フラグ

- : 影響受けない
- 0 : リセット
- 1 : セット
- × : 不定
- ↓ : 演算結果に従った影響を受ける
- P : "1" 偶数パリティ, "0" 奇数パリティ
- V : "1" オーバフロー有り, "0" オーバフロー無し
- IFF : P/Vフラグ ← (IFF)
- ① BC-1=0 ならば P/V=0, その他 P/V=1
- ② A=(HL) ならば Z=1, その他 Z=0
- ③ B-1=0 ならば Z=1, その他 Z=0

2. μ COM-82 機械語 \leftrightarrow ニーモニック対応表

| 機 械 語 \longleftrightarrow ニーモニック | | | |
|------------------------------------|---------------|----------------|--|
| 00 NOP | 40 LD B, B | 80 ADD A, B | C0 RET NZ |
| 01 LD BC, nn | 41 LD B, C | 81 ADD A, C | C1 POP BC |
| 02 LD (BC), A | 42 LD B, D | 82 ADD A, D | C2 JP NZ, nn |
| 03 INC BC | 43 LD B, E | 83 ADD A, E | C3 JP nn |
| 04 INC B | 44 LD B, H | 84 ADD A, H | C4 CALL NZ, nn |
| 05 DEC B | 45 LD B, L | 85 ADD A, L | C5 PUSH BC |
| 06 LD B, n | 46 LD B, (HL) | 86 ADD A, (HL) | C6 ADD A, n |
| 07 RLCA | 47 LD B, A | 87 ADD A, A | C7 RST 00H |
| 08 EX AF, AF | 48 LD C, B | 88 ADC A, B | C8 RET Z |
| 09 ADD HL, BC | 49 LD C, C | 89 ADC A, C | C9 RET |
| 0A LD A, (BC) | 4A LD C, D | 8A ADC A, D | CA JP Z, nn |
| 0B DEC BC | 4B LD C, E | 8B ADC A, E | CB |
| 0C INC C | 4C LD C, H | 8C ADC A, H | CC CALL Z, nn |
| 0D DEC C | 4D LD C, L | 8D ADC A, L | CD CALL nn |
| 0E LD C, n | 4E LD C, (HL) | 8E ADC A, (HL) | CE ADC A, n |
| 0F RRCA | 4F LD C, A | 8F ADC A, A | CF RST 08H |
| 10 DJNZ e | 50 LD D, B | 90 SUB B | D0 RET NC |
| 11 LD DE, nn | 51 LD D, C | 91 SUB C | D1 POP DE |
| 12 LD (DE), A | 52 LD D, D | 92 SUB D | D2 JP NC, nn |
| 13 INC DE | 53 LD D, E | 93 SUB E | D3 OUT n, A |
| 14 INC D | 54 LD D, H | 94 SUB H | D4 CALL NC, nn |
| 15 DEC D | 55 LD D, L | 95 SUB L | D5 PUSH DE |
| 16 LD D, n | 56 LD D, (HL) | 96 SUB (HL) | D6 SUB n |
| 17 RLA | 57 LD D, A | 97 SUB A | D7 RST 10H |
| 18 JR e | 58 LD E, B | 98 SBC A, B | D8 RET C |
| 19 ADD HL, DE | 59 LD E, C | 99 SBC A, C | D9 EXX |
| 1A LD A, (DE) | 5A LD E, D | 9A SBC A, D | DA JP C, nn |
| 1B DEC DE | 5B LD E, E | 9B SBC A, E | DB IN A, n |
| 1C INC E | 5C LD E, H | 9C SBC A, H | DC CALL C, nn |
| 1D DEC E | 5D LD E, L | 9D SBC A, L | DD |
| 1E LD E, n | 5E LD E, (HL) | 9E SBC A, (HL) | DE SBC A, n |
| 1F RRA | 5F LD E, A | 9F SBC A, A | DF RST 18H |
| 20 JR NZ, e | 60 LD H, B | A0 AND B | E0 RET PO |
| 21 LD HL, nn | 61 LD H, C | A1 AND C | E1 POP HL |
| 22 LD (nn), HL | 62 LD H, D | A2 AND D | E2 JP PO, nn |
| 23 INC HL | 63 LD H, E | A3 AND E | E3 EX (SP), HL |
| 24 INC H | 64 LD H, H | A4 AND H | E4 CALL PO, nn |
| 25 DEC H | 65 LD H, L | A5 AND L | E5 PUSH HL |
| 26 LD H, n | 66 LD H, (HL) | A6 AND (HL) | E6 AND n |
| 27 DAA | 67 LD H, A | A7 AND A | E7 RST 20H |
| 28 JR Z, e | 68 LD L, B | A8 XOR B | E8 RET PE |
| 29 ADD HL, HL | 69 LD L, C | A9 XOR C | E9 JP (HL) |
| 2A LD HL, (nn) | 6A LD L, D | AA XOR D | EA JP PE, nn |
| 2B DEC HL | 6B LD L, E | AB XOR E | EB EX DE, HL |
| 2C INC L | 6C LD L, H | AC XOR H | EC CALL PE, nn |
| 2D DEC L | 6D LD L, L | AD XOR L | ED |
| 2E LD L, n | 6E LD L, (HL) | AE XOR (HL) | EE XOR n |
| 2F CPL | 6F LD L, A | AF XOR A | EF RST 28H |
| 30 JR NC, e | 70 LD (HL), B | B0 OR B | F0 RET P |
| 31 LD SP, nn | 71 LD (HL), C | B1 OR C | F1 POP AF |
| 32 LD (nn), A | 72 LD (HL), D | B2 OR D | F2 JP P, nn |
| 33 INC SP | 73 LD (HL), E | B3 OR E | F3 DI |
| 34 INC (HL) | 74 LD (HL), H | B4 OR H | F4 CALL P, nn |
| 35 DEC (HL) | 75 LD (HL), L | B5 OR L | F5 PUSH AF |
| 36 LD (HL), n | 76 HALT | B6 OR (HL) | F6 OR n |
| 37 SCF | 77 LD (HL), A | B7 OR A | F7 RST 30H |
| 38 JR C, e | 78 LD A, B | B8 CP B | F8 RET M |
| 39 ADD HL, SP | 79 LD A, C | B9 CP C | F9 LD SP, HL |
| 3A LD A, (nn) | 7A LD A, D | BA CP D | FA JP M, nn |
| 3B DEC SP | 7B LD A, E | BB CP E | FB EI |
| 3C INC A | 7C LD A, H | BC CP H | FC CALL M, nn |
| 3D DEC A | 7D LD A, L | BD CP L | FD |
| 3E LD A, n | 7E LD A, (HL) | BE CP (HL) | FE CP n |
| 3F CCF | 7F LD A, A | BF CP A | FF RST 38H |

| C B ×× | | | | | | | | | | | |
|-----------|-----|------|----|-----|---------|----|-----|---------|----|-----|---------|
| 00 | RLC | B | 40 | BIT | 0, B | 80 | RES | 0, B | C0 | SET | 0, B |
| 01 | RLC | C | 41 | BIT | 0, C | 81 | RES | 0, C | C1 | SET | 0, C |
| 02 | RLC | D | 42 | BIT | 0, D | 82 | RES | 0, D | C2 | SET | 0, D |
| 03 | RLC | E | 43 | BIT | 0, E | 83 | RES | 0, E | C3 | SET | 0, E |
| 04 | RLC | H | 44 | BIT | 0, H | 84 | RES | 0, H | C4 | SET | 0, H |
| 05 | RLC | L | 45 | BIT | 0, L | 85 | RES | 0, L | C5 | SET | 0, L |
| 06 | RLC | (HL) | 46 | BIT | 0, (HL) | 86 | RES | 0, (HL) | C6 | SET | 0, (HL) |
| 07 | RLC | A | 47 | BIT | 0, A | 87 | RES | 0, A | C7 | SET | 0, A |
| 08 | RRC | B | 48 | BIT | 1, B | 88 | RES | 1, B | C8 | SET | 1, B |
| 09 | RRC | C | 49 | BIT | 1, C | 89 | RES | 1, C | C9 | SET | 1, C |
| 0A | RRC | D | 4A | BIT | 1, D | 8A | RES | 1, D | CA | SET | 1, D |
| 0B | RRC | E | 4B | BIT | 1, E | 8B | RES | 1, E | CB | SET | 1, E |
| 0C | RRC | H | 4C | BIT | 1, H | 8C | RES | 1, H | CC | SET | 1, H |
| 0D | RRC | L | 4D | BIT | 1, L | 8D | RES | 1, L | CD | SET | 1, L |
| 0E | RRC | (HL) | 4E | BIT | 1, (HL) | 8E | RES | 1, (HL) | CE | SET | 1, (HL) |
| 0F | RRC | A | 4F | BIT | 1, A | 8F | RES | 1, A | CF | SET | 1, A |
| | | | | | | | | | | | |
| 10 | RL | B | 50 | BIT | 2, B | 90 | RES | 2, B | D0 | SET | 2, B |
| 11 | RL | C | 51 | BIT | 2, C | 91 | RES | 2, C | D1 | SET | 2, C |
| 12 | RL | D | 52 | BIT | 2, D | 92 | RES | 2, D | D2 | SET | 2, D |
| 13 | RL | E | 53 | BIT | 2, E | 93 | RES | 2, E | D3 | SET | 2, E |
| 14 | RL | H | 54 | BIT | 2, H | 94 | RES | 2, H | D4 | SET | 2, H |
| 15 | RL | L | 55 | BIT | 2, L | 95 | RES | 2, L | D5 | SET | 2, L |
| 16 | RL | (HL) | 56 | BIT | 2, (HL) | 96 | RES | 2, (HL) | D6 | SET | 2, (HL) |
| 17 | RL | A | 57 | BIT | 2, A | 97 | RES | 2, A | D7 | SET | 2, A |
| 18 | RR | B | 58 | BIT | 3, B | 98 | RES | 3, B | D8 | SET | 3, B |
| 19 | RR | C | 59 | BIT | 3, C | 99 | RES | 3, C | D9 | SET | 3, C |
| 1A | RR | D | 5A | BIT | 3, D | 9A | RES | 3, D | DA | SET | 3, D |
| 1B | RR | E | 5B | BIT | 3, E | 9B | RES | 3, E | DB | SET | 3, E |
| 1C | RR | H | 5C | BIT | 3, H | 9C | RES | 3, H | DC | SET | 3, H |
| 1D | RR | L | 5D | BIT | 3, L | 9D | RES | 3, L | DD | SET | 3, L |
| 1E | RR | (HL) | 5E | BIT | 3, (HL) | 9E | RES | 3, (HL) | DE | SET | 3, (HL) |
| 1F | RR | A | 5F | BIT | 3, A | 9F | RES | 3, A | DF | SET | 3, A |
| | | | | | | | | | | | |
| 20 | SLA | B | 60 | BIT | 4, B | A0 | RES | 4, B | E0 | SET | 4, B |
| 21 | SLA | C | 61 | BIT | 4, C | A1 | RES | 4, C | E1 | SET | 4, C |
| 22 | SLA | D | 62 | BIT | 4, D | A2 | RES | 4, D | E2 | SET | 4, D |
| 23 | SLA | E | 63 | BIT | 4, E | A3 | RES | 4, E | E3 | SET | 4, E |
| 24 | SLA | H | 64 | BIT | 4, H | A4 | RES | 4, H | E4 | SET | 4, H |
| 25 | SLA | L | 65 | BIT | 4, L | A5 | RES | 4, L | E5 | SET | 4, L |
| 26 | SLA | (HL) | 66 | BIT | 4, (HL) | A6 | RES | 4, (HL) | E6 | SET | 4, (HL) |
| 27 | SLA | A | 67 | BIT | 4, A | A7 | RES | 4, A | E7 | SET | 4, A |
| 28 | SRA | B | 68 | BIT | 5, B | A8 | RES | 5, B | E8 | SET | 5, B |
| 29 | SRA | C | 69 | BIT | 5, C | A9 | RES | 5, C | E9 | SET | 5, C |
| 2A | SRA | D | 6A | BIT | 5, D | AA | RES | 5, D | EA | SET | 5, D |
| 2B | SRA | E | 6B | BIT | 5, E | AB | RES | 5, E | EB | SET | 5, E |
| 2C | SRA | H | 6C | BIT | 5, H | AC | RES | 5, H | EC | SET | 5, H |
| 2D | SRA | L | 6D | BIT | 5, L | AD | RES | 5, L | ED | SET | 5, L |
| 2E | SRA | (HL) | 6E | BIT | 5, (HL) | AE | RES | 5, (HL) | EE | SET | 5, (HL) |
| 2F | SRA | A | 6F | BIT | 5, A | AF | RES | 5, A | EF | SET | 5, A |
| | | | | | | | | | | | |
| 30 | | | 70 | BIT | 6, B | B0 | RES | 6, B | F0 | SET | 6, B |
| 31 | | | 71 | BIT | 6, C | B1 | RES | 6, C | F1 | SET | 6, C |
| 32 | | | 72 | BIT | 6, D | B2 | RES | 6, D | F2 | SET | 6, D |
| 33 | | | 73 | BIT | 6, E | B3 | RES | 6, E | F3 | SET | 6, E |
| 34 | | | 74 | BIT | 6, H | B4 | RES | 6, H | F4 | SET | 6, H |
| 35 | | | 75 | BIT | 6, L | B5 | RES | 6, L | F5 | SET | 6, L |
| 36 | | | 76 | BIT | 6, (HL) | B6 | RES | 6, (HL) | F6 | SET | 6, (HL) |
| 37 | | | 77 | BIT | 6, A | B7 | RES | 6, A | F7 | SET | 6, A |
| 38 | SRL | B | 78 | BIT | 7, B | B8 | RES | 7, B | F8 | SET | 7, B |
| 39 | SRL | C | 79 | BIT | 7, C | B9 | RES | 7, C | F9 | SET | 7, C |
| 3A | SRL | D | 7A | BIT | 7, D | BA | RES | 7, D | FA | SET | 7, D |
| 3B | SRL | E | 7B | BIT | 7, E | BB | RES | 7, E | FB | SET | 7, E |
| 3C | SRL | H | 7C | BIT | 7, H | BC | RES | 7, H | FC | SET | 7, H |
| 3D | SRL | L | 7D | BIT | 7, L | BD | RES | 7, L | FD | SET | 7, L |
| 3E | SRL | (HL) | 7E | BIT | 7, (HL) | BE | RES | 7, (HL) | FE | SET | 7, (HL) |
| 3F | SRL | A | 7F | BIT | 7, A | BF | RES | 7, A | FF | SET | 7, A |

| D D × × | | | E D × × | | | F D × × | | |
|----------|------|-----------|---------|--------|----------|----------|------|-----------|
| 0 9 | ADD | IX, BC | 4 0 | IN | B, (C) | 0 9 | ADD | IY, BC |
| 1 9 | ADD | IX, DE | 4 1 | OUT | (C), B | 1 9 | ADD | IY, DE |
| 2 1 | LD | IX, nn | 4 2 | SBC | HL, BC | 2 1 | LD | IY, nn |
| 2 2 | LD | (nn), IX | 4 3 | LD | (nn), BC | 2 2 | LD | (nn), IY |
| 2 3 | INC | IX | 4 4 | NEG | | 2 3 | INC | IY |
| 2 9 | ADD | IX, IX | 4 5 | RETN | | 2 9 | ADD | IY, IY |
| 2 A | LD | IX, (nn) | 4 6 | IM | 0 | 2 A | LD | IY, (nn) |
| 2 B | DEC | IX | 4 7 | LD | I, A | 2 B | DEC | IY |
| 3 4 | INC | (IX+d) | 4 8 | IN | C, (C) | 3 4 | INC | (IY+d) |
| 3 5 | DEC | (IX+d) | 4 9 | OUT | (C), C | 3 5 | DEC | (IY+d) |
| 3 6 | LD | (IX+d), n | 4 A | ADC | HL, BC | 3 6 | LD | (IY+d), n |
| 3 9 | ADD | IX, SP | 4 B | LD | BC, (nn) | 3 9 | ADD | IY, SP |
| 4 6 | LD | B, (IX+d) | 4 D | RET I | | 4 6 | LD | B, (IY+d) |
| 4 E | LD | C, (IX+d) | 4 F | LD | R, A | 4 E | LD | C, (IY+d) |
| 5 6 | LD | D, (IX+d) | 5 0 | IN | D, (C) | 5 6 | LD | D, (IY+d) |
| 5 E | LD | E, (IX+d) | 5 1 | OUT | (C), D | 5 E | LD | E, (IY+d) |
| 6 6 | LD | H, (IX+d) | 5 2 | SBC | HL, DE | 6 6 | LD | H, (IY+d) |
| 6 E | LD | L, (IX+d) | 5 3 | LD | (nn), DE | 6 E | LD | L, (IY+d) |
| 7 0 | LD | (IX+d), B | 5 6 | IM | 1 | 7 0 | LD | (IY+d), B |
| 7 1 | LD | (IX+d), C | 5 7 | LD | A, I | 7 1 | LD | (IY+d), C |
| 7 2 | LD | (IX+d), D | 5 8 | IN | E, (C) | 7 2 | LD | (IY+d), D |
| 7 3 | LD | (IX+d), E | 5 9 | OUT | (C), E | 7 3 | LD | (IY+d), E |
| 7 4 | LD | (IX+d), H | 5 A | ADC | HL, DE | 7 4 | LD | (IY+d), H |
| 7 5 | LD | (IX+d), L | 5 B | LD | DE, (nn) | 7 5 | LD | (IY+d), L |
| 7 7 | LD | (IX+d), A | 5 E | IM | 2 | 7 7 | LD | (IY+d), A |
| 7 E | LD | A, (IX+d) | 5 F | LD | A, R | 7 E | LD | A, (IY+d) |
| 8 6 | ADD | A, (IX+d) | 6 0 | IN | H, (C) | 8 6 | ADD | A, (IY+d) |
| 8 E | ADC | A, (IX+d) | 6 1 | OUT | (C), H | 8 E | ADC | A, (IY+d) |
| 9 6 | SUB | (IX+d) | 6 2 | SBC | HL, HL | 9 6 | SUB | (IY+d) |
| 9 E | SBC | A, (IX+d) | 6 7 | RRD | | 9 E | SBC | A, (IY+d) |
| A 6 | AND | (IX+d) | 6 8 | IN | L, (C) | A 6 | AND | (IY+d) |
| A E | XOR | (IX+d) | 6 9 | OUT | (C), L | A E | XOR | (IY+d) |
| B 6 | OR | (IX+d) | 6 A | ADC | HL, HL | B 6 | OR | (IY+d) |
| B E | CP | (IX+d) | 6 F | RLD | | B E | CP | (IY+d) |
| CB d 0 6 | RLC | (IX+d) | 7 2 | SBC | HL, SP | CB d 0 6 | RLC | (IY+d) |
| CB d 0 E | RRC | (IX+d) | 7 3 | LD | (nn), SP | CB d 0 E | RRC | (IY+d) |
| CB d 1 6 | RL | (IX+d) | 7 8 | IN | A, (C) | CB d 1 6 | RL | (IY+d) |
| CB d 1 E | RR | (IX+d) | 7 9 | OUT | (C), A | CB d 1 E | RR | (IY+d) |
| CB d 2 6 | SLA | (IX+d) | 7 A | ADC | HL, SP | CB d 2 6 | SLA | (IY+d) |
| CB d 2 E | SRA | (IX+d) | 7 B | LD | SP, (nn) | CB d 2 E | SRA | (IY+d) |
| CB d 3 E | SRL | (IX+d) | A 0 | LD I | | CB d 3 E | SRL | (IY+d) |
| CB d 4 6 | BIT | 0, (IX+d) | A 1 | CPI | | CB d 4 6 | BIT | 0, (IY+d) |
| CB d 4 E | BIT | 1, (IX+d) | A 2 | INI | | CB d 4 E | BIT | 1, (IY+d) |
| CB d 5 6 | BIT | 2, (IX+d) | A 3 | OUTI | | CB d 5 6 | BIT | 2, (IY+d) |
| CB d 5 E | BIT | 3, (IX+d) | A 8 | LDD | | CB d 5 E | BIT | 3, (IY+d) |
| CB d 6 6 | BIT | 4, (IX+d) | A 9 | CPD | | CB d 6 6 | BIT | 4, (IY+d) |
| CB d 6 E | BIT | 5, (IX+d) | AA | IND | | CB d 6 E | BIT | 5, (IY+d) |
| CB d 7 6 | BIT | 6, (IX+d) | AB | OUTD | | CB d 7 6 | BIT | 6, (IY+d) |
| CB d 7 E | BIT | 7, (IX+d) | B 0 | LD I R | | CB d 7 E | BIT | 7, (IY+d) |
| CB d 8 6 | RES | 0, (IX+d) | B 1 | CPI R | | CB d 8 6 | RES | 0, (IY+d) |
| CB d 8 E | RES | 1, (IX+d) | B 2 | IN I R | | CB d 8 E | RES | 1, (IY+d) |
| CB d 9 6 | RES | 2, (IX+d) | B 3 | OT I R | | CB d 9 6 | RES | 2, (IY+d) |
| CB d 9 E | RES | 3, (IX+d) | B 8 | LDDR | | CB d 9 E | RES | 3, (IY+d) |
| CB d A 6 | RES | 4, (IX+d) | B 9 | CPDR | | CB d A 6 | RES | 4, (IY+d) |
| CB d A E | RES | 5, (IX+d) | BA | INDR | | CB d A E | RES | 5, (IY+d) |
| CB d B 6 | RES | 6, (IX+d) | BB | OTDR | | CB d B 6 | RES | 6, (IY+d) |
| CB d B E | RES | 7, (IX+d) | | | | CB d B E | RES | 7, (IY+d) |
| CB d C 6 | SET | 0, (IX+d) | | | | CB d C 6 | SET | 0, (IY+d) |
| CB d C E | SET | 1, (IX+d) | | | | CB d C E | SET | 1, (IY+d) |
| CB d D 6 | SET | 2, (IX+d) | | | | CB d D 6 | SET | 2, (IY+d) |
| CB d D E | SET | 3, (IX+d) | | | | CB d D E | SET | 3, (IY+d) |
| CB d E 6 | SET | 4, (IX+d) | | | | CB d E 6 | SET | 4, (IY+d) |
| CB d E E | SET | 5, (IX+d) | | | | CB d E E | SET | 5, (IY+d) |
| CB d F 6 | SET | 6, (IX+d) | | | | CB d F 6 | SET | 6, (IY+d) |
| CB d F E | SET | 7, (IX+d) | | | | CB d F E | SET | 7, (IY+d) |
| E 1 | POP | IX | | | | E 1 | POP | IY |
| E 3 | EX | (SP), IX | | | | E 3 | EX | (SP), IY |
| E 5 | PUSH | IX | | | | E 5 | PUSH | IY |
| E 9 | JP | (IX) | | | | E 9 | JP | (IY) |
| F 9 | LD | SP, IX | | | | F 9 | LD | SP, IY |

3. μ COM-82 ニーモニック \longleftrightarrow 機械語 対照表

8ビット・ロード

| \times | I | R | A | B | C | D | E | H | L | (HL) | (BC) | (DE) | (IX+d) | (IY+d) | (nn) | n |
|---------------------|-------|-------|------------|------------|------------|------------|------------|------------|------------|------|------|------|------------|------------|----------|-------------|
| LD A, \times | ED 57 | ED 5F | 7F | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 0A | 1A | DD 7E d | FD 7E d | 3A nn | 3E n |
| LD B, \times | | | 47 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | | | DD 46 d | FD 46 d | | 06 n |
| LD C, \times | | | 4F | 48 | 49 | 4A | 4B | 4C | 4D | 4E | | | DD 4E d | FD 4E d | | 0E n |
| LD D, \times | | | 57 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | | | DD 56 d | FD 56 d | | 16 n |
| LD E, \times | | | 5F | 58 | 59 | 5A | 5B | 5C | 5D | 5E | | | DD 5E d | FD 5E d | | 1E n |
| LD H, \times | | | 67 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | | | DD 66 d | FD 66 d | | 26 n |
| LD L, \times | | | 6F | 68 | 69 | 6A | 6B | 6C | 6D | 6E | | | DD 6E d | FD 6E d | | 2E n |
| LD (HL), \times | | | 77 | 70 | 71 | 72 | 73 | 74 | 75 | | | | | | | 36 n |
| LD (BC), \times | | | 02 | | | | | | | | | | | | | |
| LD (DE), \times | | | 12 | | | | | | | | | | | | | |
| LD (IX+d), \times | | | DD 77 d | DD 70 d | DD 71 d | DD 72 d | DD 73 d | DD 74 d | DD 75 d | | | | | | | DD 36 dn |
| LD (IY+d), \times | | | FD 77 d | FD 70 d | FD 71 d | FD 72 d | FD 73 d | FD 74 d | FD 75 d | | | | | | | FD 36 dn |
| LD (nn), \times | | | 32 nn | | | | | | | | | | | | | |
| LD I, \times | | | ED 47 | | | | | | | | | | | | | |
| LD R, \times | | | ED 4F | | | | | | | | | | | | | |

16ビット・ロード

| \times | AF | BC | DE | HL | SP | IX | IY | nn | (nn) |
|-------------------|----|-------------|-------------|----------|-------------|-------------|-------------|-------------|-------------|
| LD AF, \times | | | | | | | | | |
| LD BC, \times | | | | | | | | 01 nn | ED 4B nn |
| LD DE, \times | | | | | | | | 11 nn | ED 5B nn |
| LD HL, \times | | | | | | | | 21 nn | 2A nn |
| LD SP, \times | | | | F9 | | DD F9 | FD F9 | 31 nn | ED 7B nn |
| LD IX, \times | | | | | | | | DD 21 nn | DD 2A nn |
| LD IY, \times | | | | | | | | FD 21 nn | FD 2A nn |
| LD (nn), \times | | ED 43 nn | ED 53 nn | 22 nn | ED 73 nn | DD 22 nn | FD 22 nn | | |
| PUSH \times | F5 | C5 | D5 | E5 | | DD E5 | FD E5 | | |
| POP \times | F1 | C1 | D1 | E1 | | DD E1 | FD E1 | | |

ブロック転送

| | |
|------|-------|
| LDI | ED A0 |
| LDIR | ED B0 |
| LDD | ED A8 |
| LDDR | ED B8 |

ブロック・サーチ

| | |
|------|-------|
| CPI | ED A1 |
| CPIR | ED B1 |
| CPD | ED A9 |
| CPDR | ED B9 |

8ビット算術論理演算

| × | A | B | C | D | E | H | L | (HL) | (IX +d) | (IY +d) | n |
|----------|----|----|----|----|----|----|----|------|---------------|---------------|---------|
| ADD A, × | 87 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | DD 86 d | FD 86 d | C6 n |
| ADC A, × | 8F | 88 | 89 | 8A | 8B | 8C | 8D | 8E | DD 8E d | FD 8E d | CE n |
| SUB × | 97 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | DD 96 d | FD 96 d | D6 n |
| SBC A, × | 9F | 98 | 99 | 9A | 9B | 9C | 9D | 9E | DD 9E d | FD 9E d | DE n |
| AND × | A7 | A0 | A1 | A2 | A3 | A4 | A5 | A6 | DD A6 d | FD A6 d | E6 n |
| XOR × | AF | A8 | A9 | AA | AB | AC | AD | AE | DD AE d | FD AE d | EE n |
| OR × | B7 | B0 | B1 | B2 | B3 | B4 | B5 | B6 | DD B6 d | FD B6 d | F6 n |
| CP × | BF | B8 | B9 | BA | BB | BC | BD | BE | DD BE d | FD BE d | FE n |
| INC × | 3C | 04 | 0C | 14 | 1C | 24 | 2C | 34 | DD 34 d | FD 34 d | |
| DEC × | 3D | 05 | 0D | 15 | 1D | 25 | 2D | 35 | DD 35 d | FD 35 d | |

CPUコントロール

| | |
|------|----------|
| NOP | 00 |
| HALT | 76 |
| DI | F3 |
| EI | FB |
| IM 0 | ED 46 |
| IM 1 | ED 56 |
| IM 2 | ED 5E |

16ビット算術演算

| × | BC | DE | HL | SP | IX | IY |
|-----------|----------|----------|----------|----------|----------|----------|
| ADD HL, × | 09 | 19 | 29 | 39 | | |
| ADD IX, × | DD 09 | DD 19 | | DD 39 | DD 29 | |
| ADD IY, × | FD 09 | FD 19 | | FD 39 | | FD 29 |
| ADC HL, × | ED 4A | ED 5A | ED 6A | ED 7A | | |
| SBC HL, × | ED 42 | ED 52 | ED 62 | ED 72 | | |
| INC × | 03 | 13 | 23 | 33 | DD 23 | FD 23 |
| DEC × | 0B | 1B | 2B | 3B | DD 2B | FD 2B |

エクスチェンジ

| | |
|-------------|----------|
| EX AF, AF' | 08 |
| EX DE, HL | EB |
| EX (SP), HL | E3 |
| EX (SP), IX | DD E3 |
| EX (SP), IY | FD E3 |
| EXX | D9 |

アキュムレータ操作

| | |
|-----|----------|
| DAA | 27 |
| CPL | 2F |
| NEG | ED 44 |
| CCF | 3F |
| SCF | 37 |

ローテート、シフト

| × | A | B | C | D | E | H | L | (HL) | (IX +d) | (IY +d) |
|-------|----------|----------|----------|----------|----------|----------|----------|----------|-----------------|-----------------|
| RLC × | CB 07 | CB 00 | CB 01 | CB 02 | CB 03 | CB 04 | CB 05 | CB 06 | DD CB d06 | FD CB d06 |
| RRC × | CB 0F | CB 08 | CB 09 | CB 0A | CB 0B | CB 0C | CB 0D | CB 0E | DD CB d0E | FD CB d0E |
| RL × | CB 17 | CB 10 | CB 11 | CB 12 | CB 13 | CB 14 | CB 15 | CB 16 | DD CB d16 | FD CB d16 |
| RR × | CB 1F | CB 18 | CB 19 | CB 1A | CB 1B | CB 1C | CB 1D | CB 1E | DD CB d1E | FD CB d1E |
| SLA × | CB 27 | CB 20 | CB 21 | CB 22 | CB 23 | CB 24 | CB 25 | CB 26 | DD CB d26 | FD CB d26 |
| SRA × | CB 2F | CB 28 | CB 29 | CB 2A | CB 2B | CB 2C | CB 2D | CB 2E | DD CB d2E | FD CB d2E |
| SRL × | CB 3F | CB 38 | CB 39 | CB 3A | CB 3B | CB 3C | CB 3D | CB 3E | DD CB d3E | FD CB d3E |
| RLD | | | | | | | | ED 6F | | |
| RRD | | | | | | | | ED 67 | | |

| | A |
|------|----|
| RLCA | 07 |
| RRCA | 0F |
| RLA | 17 |
| RRA | 1F |

ジャンプ、コール、リターン

| × | UN COND | C | NC | Z | NZ | PE | PO | M | P | |
|------------|------------|-----------|-----------|-----------|-----------|----------|----------|----------|----------|-----------|
| JP ×, nn | C3 nn | DA nn | D2 nn | CA nn | C2 nn | EA nn | E2 nn | FA nn | F2 nn | |
| JR ×, e | 18 e-2 | 38 e-2 | 30 e-2 | 28 e-2 | 20 e-2 | | | | | |
| JP (HL) | E9 | | | | | | | | | |
| JP (IX) | DD E9 | | | | | | | | | |
| JP (IY) | FD E9 | | | | | | | | | |
| CALL ×, nn | CD nn | DC nn | D4 nn | CC nn | C4 nn | EC nn | E4 nn | FC nn | F4 nn | |
| DJNZ e | | | | | | | | | | 10 e-2 |
| RET × | C9 | D8 | D0 | C8 | C0 | E8 | E0 | F8 | F0 | |
| RETI | ED 4D | | | | | | | | | |
| RETN | ED 45 | | | | | | | | | |

リスタート

| | |
|---------|----|
| RST 00H | C7 |
| RST 08H | CF |
| RST 10H | D7 |
| RST 18H | DF |
| RST 20H | E7 |
| RST 28H | EF |
| RST 30H | F7 |
| RST 38H | FF |

ビット操作

| × | A | B | C | D | E | H | L | (HL) | (IX +d) | (IY +d) |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|---------------------|---------------------|
| BIT 0, × | CB 47 | CB 40 | CB 41 | CB 42 | CB 43 | CB 44 | CB 45 | CB 46 | DD CB d 46 | FD CB d 46 |
| BIT 1, × | CB 4F | CB 48 | CB 49 | CB 4A | CB 4B | CB 4C | CB 4D | CB 4E | DD CB d 4E | FD CB d 4E |
| BIT 2, × | CB 57 | CB 50 | CB 51 | CB 52 | CB 53 | CB 54 | CB 55 | CB 56 | DD CB d 56 | FD CB d 56 |
| BIT 3, × | CB 5F | CB 58 | CB 59 | CB 5A | CB 5B | CB 5C | CB 5D | CB 5E | DD CB d 5E | FD CB d 5E |
| BIT 4, × | CB 67 | CB 60 | CB 61 | CB 62 | CB 63 | CB 64 | CB 65 | CB 66 | DD CB d 66 | FD CB d 66 |
| BIT 5, × | CB 6F | CB 68 | CB 69 | CB 6A | CB 6B | CB 6C | CB 6D | CB 6E | DD CB d 6E | FD CB d 6E |
| BIT 6, × | CB 77 | CB 70 | CB 71 | CB 72 | CB 73 | CB 74 | CB 75 | CB 76 | DD CB d 76 | FD CB d 76 |
| BIT 7, × | CB 7F | CB 78 | CB 79 | CB 7A | CB 7B | CB 7C | CB 7D | CB 7E | DD CB d 7E | FD CB d 7E |
| RES 0, × | CB 87 | CB 80 | CB 81 | CB 82 | CB 83 | CB 84 | CB 85 | CB 86 | DD CB d 86 | FD CB d 86 |
| RES 1, × | CB 8F | CB 88 | CB 89 | CB 8A | CB 8B | CB 8C | CB 8D | CB 8E | DD CB d 8E | FD CB d 8E |
| RES 2, × | CB 97 | CB 90 | CB 91 | CB 92 | CB 93 | CB 94 | CB 95 | CB 96 | DD CB d 96 | FD CB d 96 |
| RES 3, × | CB 9F | CB 98 | CB 99 | CB 9A | CB 9B | CB 9C | CB 9D | CB 9E | DD CB d 9E | FD CB d 9E |
| RES 4, × | CB A7 | CB A0 | CB A1 | CB A2 | CB A3 | CB A4 | CB A5 | CB A6 | DD CB d A6 | FD CB d A6 |
| RES 5, × | CB AF | CB A8 | CB A9 | CB AA | CB AB | CB AC | CB AD | CB AE | DD CB d AE | FD CB d AE |
| RES 6, × | CB B7 | CB B0 | CB B1 | CB B2 | CB B3 | CB B4 | CB B5 | CB B6 | DD CB d B6 | FD CB d B6 |
| RES 7, × | CB BF | CB B8 | CB B9 | CB BA | CB BB | CB BC | CB BD | CB BE | DD CB d BE | FD CB d BE |
| SET 0, × | CB C7 | CB C0 | CB C1 | CB C2 | CB C3 | CB C4 | CB C5 | CB C6 | DD CB d C6 | FD CB d C6 |
| SET 1, × | CB CF | CB C8 | CB C9 | CB CA | CB CB | CB CC | CB CD | CB CE | DD CB d CE | FD CB d CE |
| SET 2, × | CB D7 | CB D0 | CB D1 | CB D2 | CB D3 | CB D4 | CB D5 | CB D6 | DD CB d D6 | FD CB d D6 |
| SET 3, × | CB DF | CB D8 | CB D9 | CB DA | CB DB | CB DC | CB DD | CB DE | DD CB d DE | FD CB d DE |
| SET 4, × | CB E7 | CB E0 | CB E1 | CB E2 | CB E3 | CB E4 | CB E5 | CB E6 | DD CB d E6 | FD CB d E6 |
| SET 5, × | CB EF | CB E8 | CB E9 | CB EA | CB EB | CB EC | CB ED | CB EE | DD CB d EE | FD CB d EE |
| SET 6, × | CB F7 | CB F0 | CB F1 | CB F2 | CB F3 | CB F4 | CB F5 | CB F6 | DD CB d F6 | FD CB d F6 |
| SET 7, × | CB FF | CB F8 | CB F9 | CB FA | CB FB | CB FC | CB FD | CB FE | DD CB d FE | FD CB d FE |

入 力

| | |
|-----------|----------|
| IN A, n | DB n |
| IN A, (C) | ED 78 |
| IN B, (C) | ED 40 |
| IN C, (C) | ED 48 |
| IN D, (C) | ED 50 |
| IN E, (C) | ED 58 |
| IN H, (C) | ED 60 |
| IN L, (C) | ED 68 |
| INI | ED A2 |
| INIR | ED B2 |
| IND | ED AA |
| INDR | ED BA |

出 力

| | |
|------------|----------|
| OUT n, A | D3 n |
| OUT (C), A | ED 79 |
| OUT (C), B | ED 41 |
| OUT (C), C | ED 49 |
| OUT (C), D | ED 51 |
| OUT (C), E | ED 59 |
| OUT (C), H | ED 61 |
| OUT (C), L | ED 69 |
| OUTI | ED A3 |
| OTIR | ED B3 |
| OUTD | ED AB |
| OTDR | ED BB |

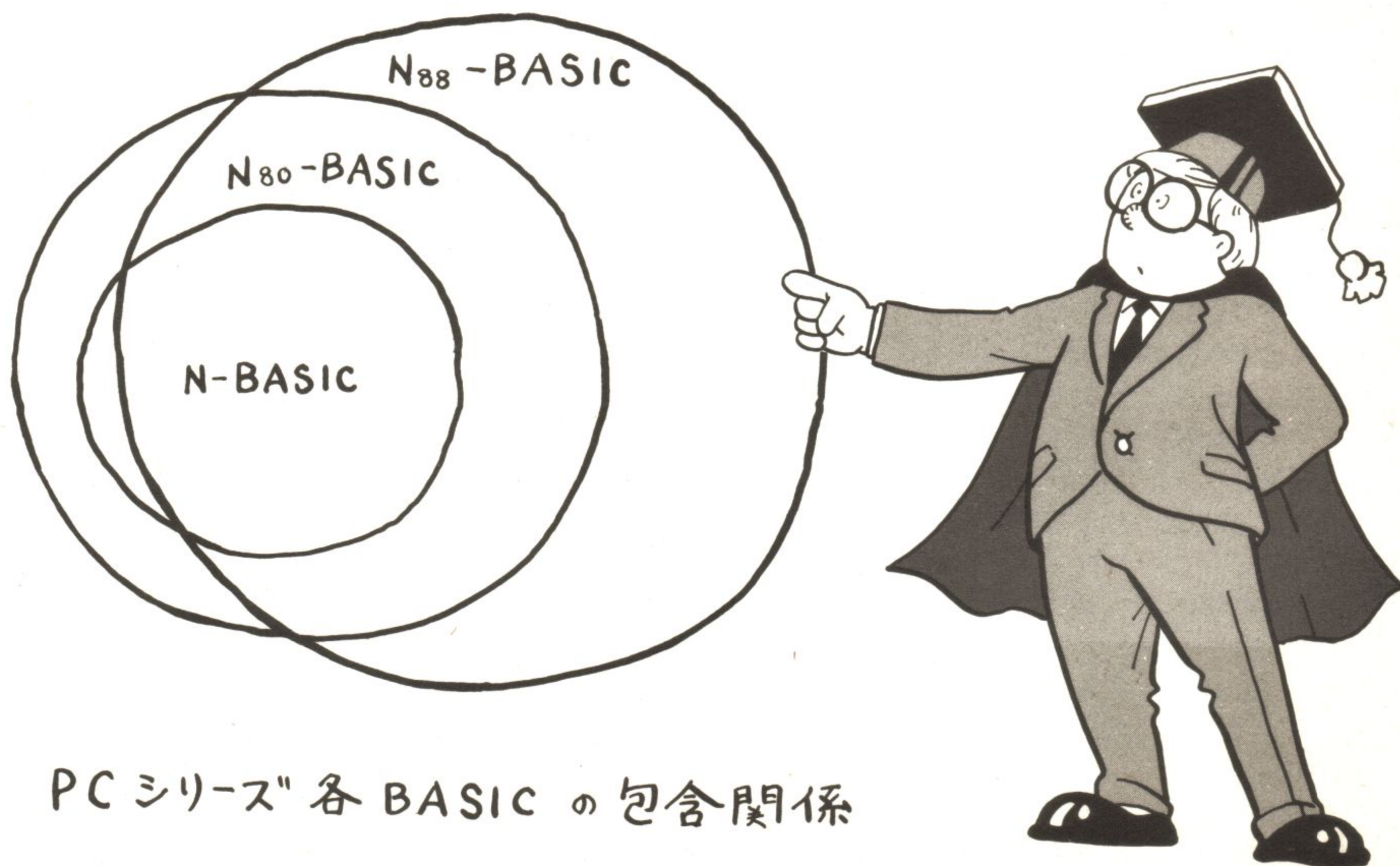
付録B N-BASIC中間言語&処理アドレス一覧表

| 命 令 | 中 間 言 語 | 処理アドレス | 命 令 | 中 間 言 語 | 処理アドレス |
|---------|---------|--------|---------|---------|--------|
| AND | F8H | | EXP | FFH+8BH | 31F3H |
| ABS | FFH+86H | 2671H | EOF | FFH+A7H | 1875H |
| ATN | FFH+8EH | 3372H | EQV | FBH | |
| ASC | FFH+95H | 5498H | FORMAT | B3H | F12FH |
| AUTO | A9H | 46CFH | FOR | 82H | 4159H |
| ATTR\$ | E7H | | FIELD | C6H | F102H |
| BCD\$ | FFH+9EH | F120H | FILES | CDH | F14DH |
| BEEP | B2H | 0D41H | FN | DCH | |
| CONSOLE | 9FH | 0884H | FRE | FFH+8FH | 5051H |
| CLOSE | CAH | F135H | FIX | FFH+A2H | 282CH |
| CONT | 99H | 4383H | FPOS | FFH+AAH | F168H |
| CLEAR | 92H | 44E8H | GOTO | 89H | 456DH |
| CLOAD | 9BH | 1F10H | GO TO | 89H | 456DH |
| CSAVE | 9AH | 1EC0H | GOSUB | 8DH | 4555H |
| CSRLIN | E6H | | GET | C7H | 1886H |
| CINT | FFH+9FH | 277FH | HEX\$ | FFH+9AH | 5275H |
| CSNG | FFH+A0H | 27B3H | INPUT | 85H | 48DAH |
| CDBL | FFH+A1H | 27DFH | IF | 8BH | 4702H |
| CVI | FFH+A3H | F0E1H | INSTR | E3H | |
| CVS | FFH+A4H | F0E4H | INT | FFH+85H | 283FH |
| CVD | FFH+A5H | F0E7H | INP | FFH+90H | 56A1H |
| COS | FFH+8CH | 32F6H | IMP | FCH | |
| CHR\$ | FFH+96H | 54A8H | INIT | D4H | 2262H |
| CMD | B8H | F0FCH | INKEY\$ | E9H | |
| COLOR | B5H | 0951H | ISSET | BDH | F105H |
| DATA | 84H | 45BEH | IRESET | BEH | F108H |
| DIM | 86H | 4E37H | IEEE | FFH+ECH | |
| DEFSTR | ABH | 445BH | KILL | CFH | F141H |
| DEFINT | ACH | 445EH | KEY | B4H | 1343H |
| DEFSNG | ADH | 4461H | LET | 88H | 45DEH |
| DEFDBL | AEH | 4464H | LOCATE | D5H | 0792H |
| DSKO\$ | C2H | F12CH | LINE | AFH | 4877H |
| DEF | 97H | 50CCH | LOAD | CBH | F138H |
| DELETE | A8H | 58D9H | LSET | D0H | F144H |
| DSKI\$ | E8H | | LPRINT | 9DH | 473AH |
| DSKF | FFH+A6H | F159H | LLIST | 9EH | 5707H |
| DEC | FFH+9DH | F123H | LPOS | FFH+9BH | 5074H |
| DATE\$ | EBH | | LISTEN | C1H | F10EH |
| END | 81H | 432FH | LIST | 93H | 570CH |
| ELSE | A1H | 45C0H | LFILS | D3H | F126H |
| ERASE | A5H | 43DFH | LOG | FFH+8AH | 2503H |
| ERROR | A6H | 46C4H | LOC | FFH+A8H | F0EDH |
| ERL | DFH | | LEN | FFH+92H | 548CH |
| ERR | E0H | | LEFT\$ | FFH+81H | 54F9H |

| 命 令 | 中 間 言 語 | 処理アドレス | 命 令 | 中 間 言 語 | 処理アドレス |
|---------|---------|--------|----------|---------|--------|
| LOF | FFH+A9H | F0F0H | SET | C9H | F156H |
| MOUNT | C4H | F153H | SAVE | D2H | F14AH |
| MERGE | CCH | F13BH | SPC (| DDH | |
| MOD | FDH | | STEP | DAH | |
| MKI\$ | FFH+ABH | F0F3H | SGN | FFH+84H | 2686H |
| MKS\$ | FFH+ACH | F0F6H | SQR | FFH+87H | 31A1H |
| MKD\$ | FFH+ADH | F0F9H | SIN | FFH+89H | 32FCH |
| MID\$ | FFH+83H | 5532H | STR\$ | FFH+93H | 527AH |
| MOTOR | B9H | 0DA1H | STRING\$ | E1H | |
| MON | B7H | 0D5DH | SPACE\$ | FFH+98H | 54DFH |
| MAT | C0H | F111H | STATUS | EEH | |
| NEXT | 83H | 4A08H | SRQ | EDH | |
| NAME | CEH | F13EH | TRON | A2H | 4396H |
| NEW | 94H | 3DE0H | TROFF | A3H | 4397H |
| NOT | DEH | | TAB (| D9H | |
| OUT | 9CH | 56ADH | TO | D7H | |
| ON | 95H | 4642H | THEN | D8H | |
| OPEN | C5H | F0FFH | TAN | FFH+8DH | 335DH |
| OR | F9H | | TERM | B6H | 0DB8H |
| OCT\$ | FFH+99H | 5270H | TALK | BFH | F10BH |
| PUT | C8H | 1891H | TIME\$ | EAH | |
| POKE | 98H | 5918H | USING | E2H | |
| PRINT | 91H | 4742H | USR | DBH | |
| POS | FFH+91H | 5079H | VAL | FFH+94H | 5553H |
| PEEK | FFH+97H | 5911H | VARPTR | E5H | |
| PORT | FFH+9CH | 20F9H | WIDTH | A0H | 0843H |
| POLL | BAH | F114H | WAIT | 96H | 56B3H |
| PSET | B1H | 06B8H | WBYTE | BCH | F117H |
| PRESET | B0H | 0705H | XOR | FAH | |
| ROINT | EFH | | + | F3H | |
| READ | 87H | 4939H | - | F4H | |
| RUN | 8AH | 453DH | * | F5H | |
| RESTORE | 8CH | 4302H | / | F6H | |
| RETURN | 8EH | 45A3H | ^ | F7H | |
| REMOVE | C3H | F150H | ¥ | FEH | |
| REM | 8FH | 45C0H | , | E4H | |
| RESUME | A7H | 468CH | > | F0H | |
| RSET | D1H | F147H | = | F1H | |
| RIGHT\$ | FFH+82H | 5529H | < | F2H | |
| RND | FFH+88H | 3283H | | | |
| RENUM | AAH | 5AEDH | | | |
| RBYTE | BBH | F11AH | | | |
| STOP | 90H | 432AH | | | |
| SWAP | A4H | 439CH | | | |

付録C N-BASIC内蔵モニタ処理アドレス一覧表

| モニタ・コマンド | | アドレス |
|----------|-------------------|-----------|
| S | (SET MEMORY) | 5 C 9 9 H |
| D | (DUMP MEMORY) | 5 D 1 6 H |
| G | (GO) | 5 D 6 8 H |
| L | (LOAD TAPE) | 5 D A E H |
| LV | (VERIFY TAPE) | 5 D A E H |
| W | (WRITE TAPE) | 5 D 7 4 H |
| TM | (TEST MEMORY) | 5 D E 6 H |
| CTRL+L | (CLEAR SCREEN) | 5 C 3 C H |
| CTRL+M | (CARRIAGE RETURN) | 5 C 6 6 H |
| CTRL+J | (LINE FEED) | 5 C 6 6 H |
| SPACE | (SPACE) | 5 C 6 6 H |
| CTRL+B | (BACK TO BASIC) | 5 C 9 3 H |
| ESC | (ESCAPE) | 5 C 6 6 H |



PCシリーズ各BASICの包含関係

付録D N-BASICシステム・サブルーチン一覧表 I

| アドレス | 機能 | レジスタ | 解 説 | アドレス | 機能 | レジスタ | 解 説 |
|-------|--------------|---------------|--|-------|--------------------|----------------|--|
| 0000H | コールド・スタート | AFBCD EHL | STOPキーの入力中であれば006AH番地のホット・スタートにジャンプし、STOPキーの入力中でなければ1757H番地にジャンプすることによりシステムのイニシャライズを行います。 システムのイニシャライズ時には、BASICのテキストを消去しますが、006AH番地のホット・スタートではBASICのテキストを保証します。 | 0257H | CRTへの1バイト出力 | ----- ---- | Aレジスタに格納されたデータ1バイトを、CRTスクリーン上の現カーソル位置に出力します。 データが20H以上の場合には通常のキャラクタとしてスクリーン上に表示しますが、20H未満のコントロール・コードであればCRTスクリーンのクリアやカーソルの移動等、コントロール・コードとしての動作を行います。 |
| 002BH | プリンタへの1バイト出力 | ----- ---- | 0D60H番地へのジャンプを行い、Aレジスタに格納されたデータ1バイトをプリンタへ出力します。 | 08F7H | テキスト・スクリーンのモード設定 | AFBCD E--- | 以下に示す各レジスタに与える入力パラメータにより、CRTのテキスト・スクリーンに関するモード設定を行います。 ●Bレジスタ……ファンクション・キー表示を行う場合にはFFHを、ファンクション・キー表示を行わない場合には00Hを与えます。 ●Cレジスタ……カラー・モードの場合にはFFHを、白黒モードの場合には00Hを与えます。 |
| 0035H | CRTへの1バイト出力 | ----- ---- | 0257H番地へのジャンプを行い、Aレジスタに格納されたデータ1バイトを、CRTスクリーン上の現カーソル位置に出力します。 データが20H以上の場合には通常のキャラクタとしてスクリーン上に表示しますが、20H未満のコントロール・コードであればCRTスクリーンのクリアやカーソルの移動等、コントロール・コードとしての動作を行います。 | 093AH | テキスト・スクリーン表示文字数の設定 | AFBCD E--- | テキスト・スクリーンに表示する最大の文字数を、Bレジスタに格納された桁数およびCレジスタに格納された行数に設定します。 通常の場合、桁数としては80/72/40/36を指定し、行数としては25/20を指定しますが、BASICでは“Illegal function call”エラーが発生するような組み合わせ、たとえば10桁×10行等も指定することができます。 |
| 006AH | ホット・スタート | AFBCD EHL | CRTスクリーン、タイマ等インターフェース関係のイニシャライズを実行後、インタプリタの制御下に入ります。ただし、BASICのテキストは保証します。 | 0D43H | 一定時間の内蔵ブザー鳴動 | A----- ---- | BASICのBEEPと同様に一定時間(約0.5秒間)内蔵ブザーを鳴動させてもどります。 すでに内蔵ブザーが鳴動中であった場合には、一定時間(約0.5秒間)内蔵ブザーの鳴動を止めてもどります。 |
| 0081H | BASICのコマンド待ち | AFBCD EHL | インターフェース関係のイニシャライズ等を全く行わずに、現在のカーソル・ポジションからプロンプト・メッセージ“Ok”を表示後、BASICのスクリーン・エディット・モードに入ります。 ただし、BASICのテキストは保証します。 0081H番地からF1B0H番地へのフックが用意されています。 | | | | |

| アドレス | 機能 | レジスタ | 解 説 | アドレス | 機能 | レジスタ | 解 説 |
|-----------|------------------|--------------------|---|-----------|------------|--------------------|---|
| 0 F 7 5 H | キーボードからの1バイト入力待ち | A F — — — — — — | キーボードからの入力認められるまで待ち、入力があった場合には、入力されたコードをAレジスタに格納してもどります。ただし、CRTスクリーンへのエコー・バックは行いません。 キーボードからの入力がない場合は入力があるまで待ち続けます。 | | | | ータと見なされます。 データの inputs は、RETURN キー (CTRL+M) または STOP キー (CTRL+C) の入力によって終了しますが、前者の場合にはキャリー・フラグ (CY) を 0 にリセットし、後者の入力によってスクリーン・エディットを中断した場合にはキャリー・フラグ (CY) を 1 にセットしてもどります。 このサブルーチンからのリターン時には、HL レジスタ対に EC 9 5 H (インプット・バッファの開始アドレス-1) が、A レジスタには最後に入力されたキャラクタのキャラクタ・コード (0 3 H または 0 D H) が与えられます。 |
| 1 B 7 E H | スクリーン・エディタ | A F B C D E H L | キーボードから、スクリーン・エディット方式によって1ラインのデータを入力し BASIC のインプット・バッファ (EC 9 6 H ~ ED 9 5 H 番地) に格納してもどります。ただし、入力データは 2 5 4 バイトまでが有効で、2 5 4 バイトを超えて入力した場合には、先頭からの 2 5 4 バイトが入力データと見なされます。 データの inputs は、RETURN キー (CTRL+M) または STOP キー (CTRL+C) の入力によって終了しますが、前者の場合にはキャリー・フラグ (CY) を 0 にリセットし、後者の入力によってスクリーン・エディットを中断した場合にはキャリー・フラグ (CY) を 1 にセットしてもどります。 このサブルーチンからのリターン時には、HL レジスタ対に EC 9 5 H (インプット・バッファの開始アドレス-1) が、A レジスタには最後に入力されたキャラクタのキャラクタ・コード (0 3 H または 0 D H) が与えられます。 | 4 0 9 5 H | レジスタ・ペアの比較 | A F — — — — — — | HL レジスタ対に格納された 1 6 ビットの無符号整数と、DE レジスタ対に格納された 1 6 ビットの無符号整数を比較し、結果をゼロ・フラグ (Z) およびキャリー・フラグ (CY) にセットしてもどります。 それぞれのフラグは、HL レジスタ対と DE レジスタ対のデータが等しい場合にはゼロ・フラグ (Z) を 1 にセットし、前者の方が小さい場合にはキャリー・フラグ (CY) を 1 にセットします。 |
| 1 B 8 A H | ライン・エディタ | A F B C D E H L | キーボードから、ライン・エディット方式によって1ラインのデータを入力し、BASIC のインプット・バッファ (EC 9 6 H ~ ED 9 5 H 番地) に格納してもどります。ただし、入力データは 2 5 4 バイトまでが有効で、2 5 4 バイトを超えて入力した場合には、先頭の 2 5 4 バイトが入力デ | 5 2 E D H | 文字列の出力 | A F B C D E H L | HL レジスタ対で指定するアドレスから 0 0 H (エンド・マーク) が格納されているアドレスまでに置かれた 2 5 5 バイト以内のデータを、EB 4 9 H 番地の出力フラグによって指定する出力デバイスへ出力します。 EB 4 9 H 番地に格納されている出力フラグの内容が、0 0 H の場合には CRT スクリーンへの出力、0 1 H ~ 7 F H の場合にはプリンタへの出力、8 0 H ~ F F H の場合には CMT への出力となります |

| アドレス | 機能 | レジスタ | 解 説 | アドレス | 機能 | レジスタ | 解 説 |
|-------|--------------------|----------------|---|-------|------------------|---------------|--|
| | | | が、通常はEB49H番地に00Hを与えてCRTテキスト・スクリーンへのメッセージ表示サブルーチンとして使用します。 | | | | それぞれのフラグは、HLレジスタ対とDEレジスタ対のデータが等しい場合にはゼロ・フラグ(Z)を1にセットし、前者の方が小さい場合にはキャリー・フラグ(CY)1にセットします。 |
| 5C66H | マシン語モニタのホット・スタート | AFBCD EHL | スタック・ポインタ(SP)の値を再設定後、マシン語モニタのプロンプト・マーク“*”を表示して、キーボードからのコマンド入力待ちに制御を移します。 | 5FB9H | 1文字入力および英大文字への変換 | AF----- | キーボードからの入力があるまで待ち、入力があった場合には、入力されたコードをAレジスタに格納後、コードが英小文字(61H~7AH)のものであれば英大文字(41H~5AH)に変換してもどります。 |
| 5E39H | 16進コード・チェック | -F----- | Aレジスタに格納されたキャラクタ・コードが16進関係でない場合にはキャリー・フラグ(CY)を1にセットし、16進関係の場合にはキャリー・フラグ(CY)を0にリセットしてもどります。 キャラクタ・コードの30H~39Hおよび41H~46Hが16進関係のコードと見なされます。 | | | | キーボードからの入力がない場合は入力があるまで待ち続けます。 エコーバック等は行われず、またSTOPキー(CTRL+C)の押下でマシン語モニタの制御下(5C66H番地)に入ります。 |
| 5E4BH | 16進コードからバイナリ形式への変換 | AF----- -HL | Aレジスタに格納された16進のキャラクタ・コードをバイナリ・コードに変換し、レジスタHLの内容を16倍(4ビット左方向にシフト)した値に加えてもどります。 このシステム・サブルーチンは、通常2~4回連続して呼び出すことによって2~4桁の16進バイナリ・コードをHLレジスタ対に生成することができます。 | 5FC1H | 英小文字から英大文字への変換 | AF----- | Aレジスタに格納されたデータが英小文字のキャラクタ・コード(61H~7AH)であれば、第5ビットをマスクして英大文字(41H~5AH)に変換します。 |
| 5EC0H | 16進4桁表示 | A----- | HLレジスタ対に格納されたデータを4桁の16進数として、CRTテキスト・スクリーン上の現カーソル位置に表示します。 | 5FCAH | CRコードおよびLFコードの表示 | A----- -F- | CRTテキスト・スクリーン上の現カーソル位置に、キャリッジ・リターン・コード(CRコード=0DH)およびライン・フィード・コード(LFコード=0AH)を出力します。 上記の出力によってカーソルは次行の先頭まで移動し、もしカーソルがスクロール範囲の最下行にあればスクロール・アップを行います。 |
| 5ED3H | レジスタ・ペアの比較 | AF----- | HLレジスタ対に格納された16ビットの無符号整数と、DEレジスタ対に格納された16ビットの無符号整数を比較し、結果をゼロ・フラグ(Z)およびキャリー・フラグ(CY)にセットしてもどります。 | 5FD4H | スペース(空白)の表示 | | CRTテキスト・スクリーン上の現カーソル位置に、スペース・コード(空白コード=20H)を出力します。 |

付録E N-BASICシステム・サブルーチン一覧表II

| アドレス | 機能 | アドレス | 機能 |
|-------|-----------------------|-------|---------------------------------|
| 0000H | コールド・スタート | 0D14H | μ PD8251ソフトウェア・リセット |
| 0008H | ホット・スタート | 0D43H | 一定時間の内蔵ブザー鳴動 |
| 0013H | ホット・スタート | 0D4BH | 内蔵ブザーのコントロール |
| 0018H | デバイスへの1バイト出力 | 0D60H | プリンタへの1バイト出力 |
| 002BH | プリンタへの1バイト出力 | 0DA3H | 内蔵リレーの反転 |
| 0035H | CRTへの1バイト出力 | 0DAEH | 内蔵リレーのコントロール |
| 006AH | ホット・スタート | 0F75H | キーボードからの1バイト入力待ち |
| 0081H | BASICのコマンド待ち | 0F7BH | キーボードからの1バイト入力 |
| 0099H | PC-8031との1セクタ入出力 | 0FACH | リアルタイム・キー・スキャニング |
| 00CBH | 拡張システム・チェック | 124AH | スクリーン・コピー |
| 0257H | CRTへの1バイト出力 | 1602H | タイマからのデータ読出し |
| 02D7H | カーソル移動とCRTへの1バイト出力 | 1663H | タイマへのデータ書込み |
| 0350H | ベルコードの出力 | 17E9H | BASICプログラム格納アドレス設定 |
| 03A9H | カーソルの移動 | 1875H | "Disk BASIC Feature" エラー |
| 03D9H | VRTCのチェック | 1B7EH | スクリーン・エディタ |
| 03F3H | キャラクタ座標からVRAMアドレスへ変換 | 1B8AH | ライン・エディタ |
| 0401H | 論理座標から絶対座標への変換 | 1F8BH | BASICテキストの終了アドレス設定 |
| 0451H | スクリーンの1ライン・クリア | 240FH | 単精度型実数の減算 |
| 045AH | スクリーンのクリア | 2412H | 単精度型実数の加算 |
| 047AH | ライン数からVRAMのバイト数を計算 | 2503H | 自然対数の計算 |
| 0487H | スクリーン・クリア・バッファの設定 | 2541H | 単精度型実数の乗算 |
| 04F8H | アトリビュートのコントロール | 259CH | 単精度型実数の除算 |
| 0664H | ライン番号からVRAMアドレスへの変換 | 267EH | 絶対値の計算 |
| 06E6H | アトリビュートのコントロール | 2676H | 実数の符号反転 |
| 07C9H | ファンクション・キー表示の開始 | 2689H | 符号の調査 |
| 08F7H | スクリーンのモード設定 | 2689H | 8ビット整数の格納 |
| 093AH | スクリーン表示文字数の設定 | 26AFH | 単精度型実数の移動1 (メモリからFACCへ) |
| 09A3H | スクリーン表示桁数の設定 | 26B2H | 単精度型実数の移動2 (EDCBレジスタからFACCへ) |
| 09D7H | スクリーン表示行数の設定 | 26BDH | 単精度型実数の移動3 (FACCからEDCBレジスタへ) |
| 09F6H | CRTの25行モード用イニシャライズ | 26C0H | 単精度型実数の移動4 (メモリからEDCBレジスタへ) |
| 0A5EH | CRTの20行モード用イニシャライズ | 26C9H | 単精度型実数の移動5 (FACCからメモリへ) |
| 0A73H | ファンクション・キー・リスト | 26D5H | 256バイト以内のブロック転送 |
| 0B18H | ブート・ストラップ・ローダ | 270CH | 単精度型実数の比較 |
| 0B2EH | スクリーン最下段の消去 | 2739H | 整数の比較 |
| 0B92H | グラフィック座標からキャラクタ座標への変換 | 2778H | 倍精度型実数の比較 |
| 0BD2H | カーソル表示の停止 | 277FH | 整数型への変換 |
| 0BE2H | カーソル表示の開始 | 278CH | 単精度実数型から整数型への変換 |
| 0BF3H | CMTからの入力イニシャライズ | 279CH | 整数の格納 |
| 0C2EH | CMTインターフェースのクローズ | 27A1H | FACCの型指定 |
| 0C46H | CMTへの出力イニシャライズ | | |
| 0C88H | CMTからの1バイト入力 | | |
| 0CB3H | アスタリスクのフラッシング | | |
| 0CDAH | CMTへの1バイト出力 | | |
| 0CF1H | STOPキー・チェック | | |

| アドレス | 機 能 | アドレス | 機 能 |
|-----------|----------------------|-----------|-----------------------|
| 2 7 B 3 H | 単精度実数型への変換 | 4 C C C H | 英小文字から英大文字への変換 2 |
| 2 7 B D H | 倍精度実数型から単精度実数型への変換 | 4 C E 9 H | 1 6 進文字列から数値データへの変換 |
| 2 7 D 0 H | 整数型から単精度実数型への変換 | 5 2 E C H | 文字列の出力 1 |
| 2 7 D F H | 倍精度実数型への変換 | 5 2 E D H | 文字列の出力 2 |
| 2 7 E 9 H | 単精度実数型から倍精度実数型への変換 | 5 C 2 C H | マシン語モニタ |
| 2 7 F 2 H | F A C C の倍精度実数型指定 | 5 C 5 E H | マシン語モニタのエラー出力 |
| 2 7 F 5 H | F A C C の単精度実数型指定 | 5 C 6 6 H | マシン語モニタのホット・スタート |
| 2 8 2 C H | 整数部の計算 | 5 D F 2 H | テスト・メモリ |
| 2 8 3 F H | 小数点以下を切り捨てた整数値の計算 | 5 E 2 1 H | 1 6 進 4 桁入力 |
| 2 8 D 2 H | 整数の減算 | 5 E 3 9 H | 1 6 進コード・チェック |
| 2 8 D D H | 整数の加算 | 5 E 4 B H | 1 6 進コードからバイナリ形式への変換 |
| 2 8 F D H | 整数の乗算 | 5 E 5 A H | アドレス・カウンタのインクリメント |
| 2 9 5 0 H | 整数の除算 1 | 5 E 8 3 H | 1 6 進数から 1 6 進コードへの変換 |
| 2 9 9 D H | 整数の符号反転 1 | 5 E 9 6 H | 4 ビット 1 6 進コードへの変換 |
| 2 9 A 0 H | 整数の符号反転 2 | 5 E A 0 H | 1 6 進コードから 1 6 進数への変換 |
| 2 9 B 2 H | 整数の剰余演算 | 5 E B 1 H | 4 ビット 1 6 進数への変換 |
| 2 9 C 3 H | 倍精度型実数の減算 | 5 E B D H | 1 6 進 2 桁表示 1 |
| 2 9 C A H | 倍精度型実数の加算 | 5 E C 0 H | 1 6 進 4 桁表示 |
| 2 A F 4 H | 倍精度型実数の乗算 | 5 E C 5 H | 1 6 進 2 桁表示 2 |
| 2 B 3 7 H | 倍精度型実数の除算 | 5 E D 3 H | レジスタ対の比較 |
| 2 B B 7 H | 文字列から倍精度実数型データへの変換 | 5 E E D H | CMT へのデータ・ブロック出力 |
| 2 B B E H | 文字列から数値データへの変換 | 5 F 2 F H | CMT への 1 バイト出力 |
| 2 D 0 B H | “ in ” と行番号の出力 | 5 F 6 A H | CMT からのデータ・ブロック入力 |
| 2 D 1 3 H | 行番号の出力 | 5 F 9 E H | CMT からの 1 バイト入力 |
| 2 D 2 2 H | 数値データから文字列への変換 | 5 F A D H | カーソル表示と英大文字入力 |
| 2 D 2 3 H | 数値から有フォーマット文字列への変換 | 5 F B 0 H | カーソル表示と 1 文字表示 |
| 3 D 9 F H | 無符号整数データから文字列への変換 | 5 F B 9 H | キーボードからの英大文字入力 |
| 3 1 A 1 H | 平方根の計算 | 5 F C 1 H | 英小文字から英大文字への変換 |
| 3 1 F 3 H | e に対する指数関数の計算 | 5 F C A H | C R コードおよび L F コードの表示 |
| 3 2 8 3 H | 乱数の発生 | 5 F D 4 H | スペース・コードの表示 |
| 3 2 F 6 H | 余弦 (コサイン) の計算 | 5 F F 2 H | V R T C チェック |
| 3 2 C 6 H | 正弦 (サイン) の計算 | | |
| 3 3 5 D H | 正接値 (タンジェント) の計算 | | |
| 3 3 7 2 H | 逆正接 (アーク・タンジェント) の計算 | | |
| 3 B F 9 H | エラー出力 | | |
| 3 D 7 6 H | B A S I C テキストのリンク 1 | | |
| 3 D 7 9 H | B A S I C テキストのリンク 2 | | |
| 3 D 7 A H | B A S I C テキストのリンク 3 | | |
| 3 D C 1 H | B A S I C の行番号サーチ | | |
| 3 E 5 C H | インプット | | |
| 4 0 9 5 H | レジスタ対の比較 | | |
| 4 0 A 6 H | デバイスへの 1 バイト出力 | | |
| 4 D E A H | 整数の除算 2 | | |
| 4 C C B H | 英小文字から英大文字への変換 1 | | |

付録F N-BASICシステム・ワークエリア一覧表

| アドレス | 内 容 | アドレス | 内 容 |
|-------|--|-------|--|
| EA00H | EA01H番地にアドレスを格納して出力ポートにAレジスタのデータを出力するサブルーチン。“OUT”で使用 | EA5CH | ファンクション・キー表示の状態。50Hのとき80桁反転 |
| EA03H | EA04H番地, EA08H番地, EA0CH番地, EA0FH番地に数値を代入してから用いる単精度型減算サブルーチン | EA5DH | スクロール範囲。0～24 |
| EA11H | 乱数用。乱数=F(直前の乱数×X+Y)のFに関する値 | EA5EH | スクロール開始行。1～25 |
| EA12H | 乱数用。下位2ビットで3種類のYを選択 | EA5FH | カーソル・スイッチ。00Hのとき消去 |
| EA13H | 乱数用。下位3ビットで8種類のXを選択 | EA60H | ファンクション・キー表示スイッチ。00Hのとき消去, FFHのとき表示 |
| EA14H | 乱数用。単精度実数型の乱数用定数8種類 | EA61H | スクリーン・モード・スイッチ。00Hのとき白黒モード, FFHのときカラー・モード |
| EA34H | 乱数用。直前に発生した値 | EA62H | テキスト・スクリーンの行数 |
| EA38H | EA39H番地にアドレスを格納して入力ポートからAレジスタにデータを入力するサブルーチン。“IN”で使用 | EA63H | カーソルの縦位置。1～25 |
| EA3BH | “USR0”～“USR9”に対応するコール先アドレスのテーブル。初期値では“Illegal function call”エラーを発生する44A5H番地を指定 | EA64H | カーソルの横位置。1～80 |
| EA4FH | 未使用 | EA65H | テキスト・スクリーンの桁数 |
| EA50H | フリー・エリアの後端アドレス。初期値はE9FFH | EA66H | 出力ポート30H番地に出力した値 |
| EA52H | 中間言語81H～D5Hに対応するテーブルの先頭アドレス。通常は33BDH | EA67H | 出力ポート40H番地に出力した値 |
| EA54H | RS-232C用。00Hのとき“Communications Buffer Overflow”エラーの発生 | EA68H | ファンクション・キー・フラグ。00H以外のときファンクション・キーが押された |
| EA55H | インタラプト・レベル。インタラプトがなければFFH | EA69H | キーボードから直前に入力されたキャラクタ・コード |
| EA56H | インタラプト発生フラグ。通常は00H | EA6AH | 特殊キーの押下状況。00Hのとき特殊キーの押下なし, 01HのときSHIFT, 02HのときCTRL, 03Hのときカナ, 04Hのときカナ+SHIFT, 05HのときGRPH |
| EA57H | 00H以外のとき現カーソル行を消去 | EA6BH | オート・リピート用のディレイ・タイム。06Hか40H |
| EA58H | ターミナル・モード・フラグ 第7ビット…ターミナル・モードの有無 第6ビット…CRとCRLFの区別 第5ビット…偶数パリティと奇数パリティの区別 第4ビット…パリティの有無 第3ビット…半二重と全二重の区別 第2ビット…ASCIIコードとJISコードの区別 第1ビット…コントロール・コード表示有無 第0ビット…エコー・バックの有無 | EA6CH | キーボード・マトリクスからの入力データ |
| EA59H | カーソル状態フラグ。00Hのとき消去, FFHのとき表示 | EA76H | BCDコードによる, 秒, 分, 時, 日, 月, 年 |
| EA5AH | スル・キャラクタ・コード。初期値は00H | EA7CH | ファンクション・キー(f・1)の定義内容 |
| EA5BH | スル・アトリビュート・コード。初期値は00H | EA8CH | ファンクション・キー(f・2)の定義内容 |
| | | EA9CH | ファンクション・キー(f・3)の定義内容 |
| | | EAACH | ファンクション・キー(f・4)の定義内容 |
| | | EABCH | ファンクション・キー(f・5)の定義内容 |
| | | EACCH | ファンクション・キー(f・6)の定義内容 |
| | | EADCH | ファンクション・キー(f・7)の定義内容 |
| | | EAECH | ファンクション・キー(f・8)の定義内容 |
| | | EAFCH | ファンクション・キー(f・9)の定義内容 |
| | | EB0CH | ファンクション・キー(f・10)の定義内容 |
| | | EB1CH | N-BASICの“PSET”, “PRESET”, “GET”, “PUT”で使用するサブルーチン条件によって変化する |
| | | EB20H | N-BASICの“LINE”で使用するサブルーチン |
| | | EB22H | N-BASICの“LINE”で使用するサブルーチン |

| アドレス | 内 容 |
|-------|---|
| EB24H | N-BASICの“PSET”, “PRESET”で使用するルーチンで、ROM内への分岐に用いる |
| EB27H | N-BASIC “LINE”で使用するルーチンで、ROM内への分岐に用いる |
| EB2AH | RS-232C-CH1用。USARTのコマンド・バイト |
| EB2BH | RS-232C-CH2用。USARTのコマンド・バイト |
| EB2CH | 未使用 |
| EB2DH | RS-232C用入力ポートのアドレス。CH1のときC1H, CH2のときC3H |
| EB2EH | RS-232C-CH1用のポインタ |
| EB2FH | RS-232C-CH2用のポインタ |
| EB30H | 未使用 |
| EB31H | RS-232Cからの入力可能バイト数 |
| EB33H | RS-232C-CH1用 |
| EB3AH | RS-232C用 |
| EB3BH | RS-232C-CH2用 |
| EB42H | RS-232C用 |
| EB43H | RS-232C-CH1への出力カウンタ |
| EB44H | RS-232C-CH2への出力カウンタ |
| EB45H | 未使用 |
| EB46H | エラー・コード。00Hのときエラー未発生 |
| EB47H | 未使用 |
| EB48H | プリンタ・ヘッドの横位置 |
| EB49H | 周辺デバイスへの出力フラグ。00HのときCRTを, 01H~7FHのときプリンタを, 80H~FFHのときCMTを, それぞれ選択 |
| EB4AH | テキスト・スクリーンの横幅 |
| EB4BH | テキスト・スクリーンの改行値 |
| EB4CH | 未使用 |
| EB4DH | ESCキー押下フラグ |
| EB4EH | BASICの“INPUT\$”で使用 |
| EB50H | フリー・エリアの終了アドレス |
| EB52H | BASICで現在実行中の行番号。ダイレクト・モードのときはFFFFH |
| EB54H | BASICのプログラム・エリア開始アドレス |
| EB56H | BASICのキーワードを中間言語に変換するために使用 |
| EC56H | バッファ |
| EC96H | インプット・バッファ。キーボードからの文字列入力に使用 |
| ED99H | カーソルの横座標 |

| アドレス | 内 容 |
|-------|---|
| ED9AH | テキスト・スクリーンの25行に対応するリンク情報 |
| EDB3H | “INPUT”で00Hのとき“Bad File Data”エラー発生, “READ”で00Hのとき“Out of data”エラー発生 |
| EDB4H | スクリーン・エディット時のカーソル横座標退避 |
| EDB5H | スクリーン・エディット時のカーソル縦座標退避 |
| EDB6H | スクリーン・エディット時のテキスト・スクリーン横幅退避 |
| EDB7H | ターミナル・モードで使用 |
| EDB9H | ディスクのセクタ番号 |
| EDBAH | セミ・グラフィック用のアトリビュート・コード |
| EDBBH | セミ・グラフィック用の縦座標 |
| EDBCH | セミ・グラフィック用の横座標 |
| EDBDH | セミ・グラフィック用のファンクション・コード |
| EDBEH | セミ・グラフィック用のVRAM (Video-RAM) アドレス |
| EDC0H | ファンクション・キー・ポインタ。現在入力中のアドレスを示す |
| EDC2H | キーボード・マトリックスの入力ポート |
| EDC3H | セミ・グラフィックのドット位置に対応するキャラクタ縦座標 |
| EDC4H | セミ・グラフィックのドット位置に対応するキャラクタ横座標 |
| EDC5H | ディスク・アクセス時のエラー・カウンタ |
| EDC6H | 指定するドライブ番号。0~3 |
| EDC7H | 接続されているドライブ数 |
| EDC8H | 未使用 |
| EDC9H | ディスク用入出力ポートの最上位アドレス |
| EDCAH | N-BASICの“LINE”で消去に使用 |
| EDCBH | ディスク・アクセス時に使用 |
| EDCCH | 未使用 |
| EDCEH | RS-232C-CH1のインプット・バッファ |
| EE4EH | RS-232C-CH2のインプット・バッファ |
| EECEH | RS-232Cへの出力データ退避用 |
| EECFH | 未使用 |
| EED0H | RS-232Cのチャンネル番号。0~1 |
| EED1H | 未使用 |
| EED2H | IEEE (2) の値 |
| EED3H | IEEE (3) の値 |
| EED4H | 未使用 |
| EED5H | IEEE-488のステータス |
| EED6H | IEEE-488関係 |
| EED7H | IEEE-488関係 |

| アドレス | 内 容 | アドレス | 内 容 |
|---------|--|-------|---|
| EED8H | IEEE-488関係 | EF3DH | 未使用 |
| EED9H | IEEE-488関係 | EF3EH | 現在CMTからロード中のファイル・ネーム |
| EEDA H | IEEE-488関係 | EF44H | 配列処理関係 |
| EEDBH | IEEE-488関係。00H以外のときイネーブル | EF45H | フローティング・アキュムレータの型, 02Hのとき整数型, 03Hのとき文字型, 04Hのとき単精度実数型, 08Hのとき倍精度実数型 |
| EEDCH | IEEE-488関係 | EF46H | “LIST”で使用 |
| EEDDH | IEEE-488関係 | EF47H | “LIST”で使用 |
| EED E H | “ON SRQ GOSUB”で指定する行番号 | EF48H | 実行中のBASICテキストのアドレス |
| EEE0H | IEEE-488関係 | EF4AH | 実行中の命令に付随するコントロール・コード。0BH~0FH, 11H~1DH, 1FH |
| EEE1H | IEEE-488関係 | EF4BH | 数値定数の型。02H, 04H, 08H |
| EEE2H | IEEE-488関係 | EF4CH | プログラム中の数値定数を1度ここに移してからフローティング・アキュムレータに格納する |
| EEE4H | IEEE-488関係 | EF54H | プログラム・エリアの終了アドレス |
| EEE6H | IEEE (6) の値 | EF56H | ストリング・ディスクリプタの格納アドレス |
| EEE7H | IEEE (4) の値 | EF58H | 11組のストリング・ディスクリプタ |
| EEE8H | IEEE (5) の値 | EF79H | フリー・エリアの終了アドレス |
| EEE9H | IEEE-488関係 | EF7BH | “DIM”, “FN”, “NEXT”で使用 |
| EEEA H | IEEE-488関係 | EF7DH | 文字型配列変数の実際のアドレス |
| EEEBH | IEEE-488関係 | EF7FH | “FOR”でテキスト・ポインタの退避用 |
| EEECH | IEEE-488関係 | EF81H | 読出し中データの行番号 |
| EEDDH | IEEE-488関係 | EF83H | “FOR”, “ERASE”, “DIM”, “FN”で使用 |
| EEEEH | “CMD DELIM”で指定されたデリミタ・コード | EF84H | “INPUT”, “READ”, “USING”で使用 |
| EEEFH | IEEE-488関係 | EF85H | “ERASE”, “FOR”, “LET”, “NEXT”で使用 |
| EEF0H | “CMD TIMEOUT”で設定されたタイム | EF87H | “GOTO”で使用 |
| EEF1H | IEEE-488関係 | EF88H | “AUTO”実行中フラグ |
| EEF3H | IEEE-488関係 | EF89H | “AUTO”カウンタ |
| EEF4H | IEEE-488関係 | EF8BH | “AUTO”増分 |
| EEF5H | IEEE-488関係 | EF8DH | BASICプログラムの先頭アドレス |
| EEF6H | 未使用 | EF8FH | “FOR”, “NEXT”, “RETURN”で使用 |
| EF2CH | 00Hのとき“PUT”, FFHのとき“GET” | EF91H | システム変数“ERL” |
| EF2DH | “GET”および“PUT”で使用 | EF93H | 直前に処理した行番号。“LIST”等の“.”で使用 |
| EF2FH | “GET”および“PUT”で使用 | EF95H | “RESUME”で使用 |
| EF31H | “GET”および“PUT”で使用 | EF97H | エラー関係 |
| EF32H | “GET”および“PUT”で使用 | EF99H | エラー発生フラグ |
| EF34H | 00Hのとき“PUT@”, FFHのとき“GET@” | EF9AH | “DIM”, “FN”, “NEXT”で使用 |
| EF35H | “GET”および“PUT”で使用 | EF9CH | ブレーク(中断)した行番号 |
| EF36H | CMTアクセス時に指定されたファイル・ネーム | EF9EH | “CONT”を行うための中断アドレス。0000Hのとき続行不可 |
| EF3CH | CMTアクセス時のエラーに対する“Tape read ERROR”エラー出力フラグ。00Hのときエラーを出力する | | |

| アドレス | 内 容 |
|-------|-------------------------------|
| EFA0H | 単純変数エリアの先頭アドレス |
| EFA2H | 配列変数エリアの先頭アドレス |
| EFA4H | 配列変数エリアの終了アドレス |
| EFA6H | 現在読出しているデータのアドレス |
| EFA8H | “DEF”によって指定された変数の型。A～Zの26バイト |
| EFC2H | “FN”で使用 |
| EFC4H | “FN”で使用 |
| EFC6H | “FN”で使用。引数の名前および処理アドレス |
| F02AH | “FN”で使用。引数の名前および処理アドレス |
| F092H | “DIM”で使用 |
| F093H | “DIM”で使用 |
| F095H | “FN”で使用 |
| F096H | ガベージ・コレクションで使用 |
| F098H | “FN”で使用 |
| F09AH | “SWAP”で使用。変数の退避用 |
| F0A2H | 00Hのとき“TROFF”, FFHのとき“TRON” |
| F0A3H | フローティング・アキュムレータの無効桁 |
| F0A4H | フローティング・アキュムレータ |
| F0ACH | フローティング・アキュムレータの符号保持 |
| F0ADH | サブ・フローティング・アキュムレータの無効桁 |
| F0AEH | サブ・フローティング・アキュムレータ |
| F0B6H | 数値データ出力用バッファ |
| F0D1H | 倍精度除算用レジスタ |
| F0D9H | 倍精度乗算用レジスタ |
| F0E1H | 命令のジャンプ・テーブル |
| F16BH | 処理のジャンプ・テーブル |
| F216H | 未使用 |
| F300H | Video-RAM |
| FEB8H | テキスト・スクリーン・クリア用のデータ 120バイト |
| FF30H | モニタの“S”コマンド用。エラー発生時のアドレス退避 |
| FF32H | 未使用 |
| FF33H | 00Hのときモニタの“L”, 0DHのときモニタの“LV” |
| FF34H | モニタ起動時のHLレジスタ対退避 |
| FF36H | モニタ起動時のスタック・ポインタ(SP)退避 |
| FF38H | 00Hのときモニタの“S”, FFHのときモニタの“D” |
| FF39H | テスト・メモリ“TM”の不良メモリのアドレス退避 |

| アドレス | 内 容 |
|-------|--------------------|
| FF3BH | テスト・メモリ“TM”の書込みデータ |
| FF3CH | テスト・メモリ“TM”の読出しデータ |
| FF3DH | 未使用 |



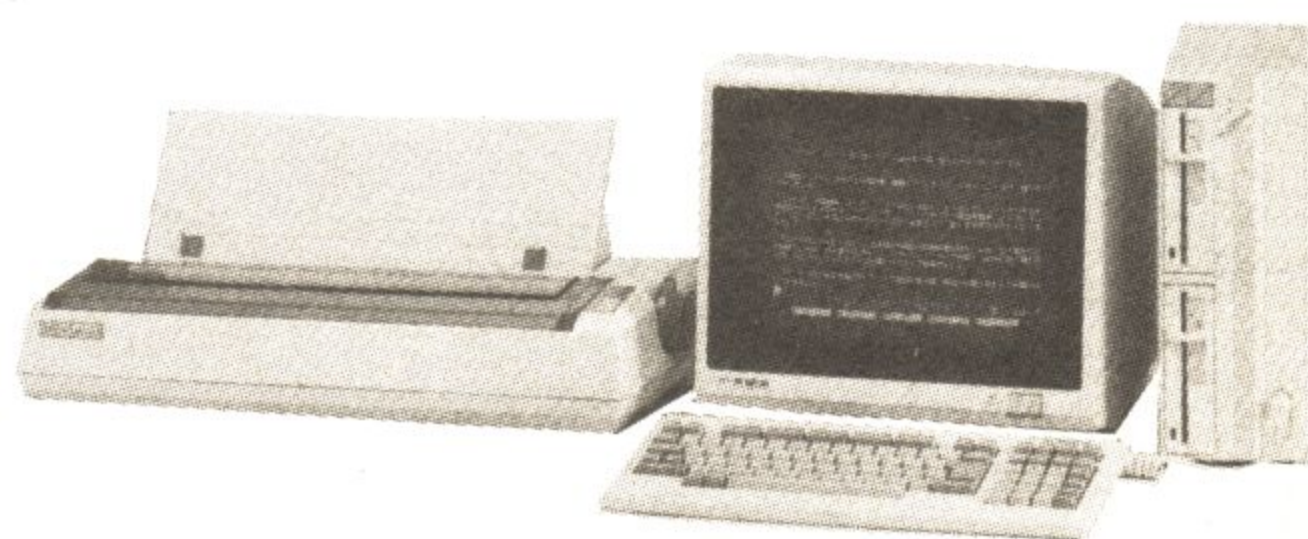
▲PC-8001シリーズ



▲PC-8001mkIIシリーズ



▲PC-8801シリーズ



▲PC-8801mkIIシリーズ

付録G μ PD3301アトリビュート・コード表

| B I T | | カ ラ ー モ ー ド | | 白 黒 モ ー ド |
|-------|---|-------------|-----------------|-----------------|
| | | カ ラ ー 指 定 | 機 能 指 定 | |
| 0 | 0 | | ノ ー マ ル | ノ ー マ ル |
| | 1 | | シ ー ク レ ッ ト | シ ー ク レ ッ ト |
| 1 | 0 | | ノ ー マ ル | ノ ー マ ル |
| | 1 | | ブ リ ン ク | ブ リ ン ク |
| 2 | 0 | | ノ ー マ ル | ノ ー マ ル |
| | 1 | | リ バ ー ス | リ バ ー ス |
| 3 | 0 | 機 能 指 定 | | |
| | 1 | カ ラ ー 指 定 | | |
| 4 | 0 | キ ャ ラ ク タ | ノ ー マ ル | ノ ー マ ル |
| | 1 | グ ラ フ ィ ッ ク | オ ー バ ー ・ ラ イ ン | オ ー バ ー ・ ラ イ ン |
| 5 | 0 | B 信 号 O F F | ノ ー マ ル | ノ ー マ ル |
| | 1 | B 信 号 O N | ア ン ダ ー ・ ラ イ ン | ア ン ダ ー ・ ラ イ ン |
| 6 | 0 | R 信 号 O F F | | |
| | 1 | R 信 号 O N | | |
| 7 | 0 | G 信 号 O F F | | キ ャ ラ ク タ |
| | 1 | G 信 号 O N | | グ ラ フ ィ ッ ク |

RGB信号⇒カラー変換表

| カラー 信号 | 黒 (ブラック) | 青 (ブルー) | 赤 (レッド) | 紫 (マゼンダ) | 緑 (グリーン) | 水 (シアン) | 黄 (イエロー) | 白 (ホワイト) |
|-----------|-------------|------------|------------|-------------|-------------|------------|-------------|-------------|
| G | OFF | OFF | OFF | OFF | ON | ON | ON | ON |
| R | OFF | OFF | ON | ON | OFF | OFF | ON | ON |
| B | OFF | ON | OFF | ON | OFF | ON | OFF | O |

付録H キャラクタ・コード表

| | | 上位4ビット→ | | | | | | | | | | | | | | | |
|---------|---|----------------|----------------|----|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | O | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 下位4ビット↓ | O | | D _E | 0 | @ | P | | p | | ⌈ | | ー | タ | ミ | ≡ | × | |
| | 1 | S _H | D ₁ | ! | I | A | Q | a | q | | | 。 | ア | チ | ム | ≡ | 円 |
| | 2 | S _X | D ₂ | " | 2 | B | R | b | r | | | 「 | イ | ツ | メ | ≡ | 年 |
| | 3 | E _X | D ₃ | # | 3 | C | S | c | s | | | 」 | ウ | テ | モ | ≡ | 月 |
| | 4 | E _T | D ₄ | \$ | 4 | D | T | d | t | | | 、 | エ | ト | ヤ | ◀ | 日 |
| | 5 | E _Q | N _K | % | 5 | E | U | e | u | | | ・ | オ | ナ | ユ | ◀ | 時 |
| | 6 | A _K | S _N | & | 6 | F | V | f | v | | | ヲ | カ | ニ | ヨ | ◀ | 分 |
| | 7 | B _L | E _B | ′ | 7 | G | W | g | w | | | ア | キ | ヌ | ラ | ◀ | 秒 |
| | 8 | B _S | C _N | (| 8 | H | X | h | x | | | イ | ク | ネ | リ | ♠ | |
| | 9 | H _T | E _M |) | 9 | I | Y | i | y | | | ウ | ケ | ノ | ル | ♥ | |
| | A | L _F | S _B | * | : | J | Z | j | z | | | エ | コ | ハ | レ | ♦ | |
| | B | H _M | E _C | + | ; | K | [| k | } | | | オ | サ | ヒ | ロ | ♣ | |
| | C | C _L | → | , | < | L | ¥ | ! | ! | | | ヤ | シ | フ | ワ | ● | |
| | D | C _R | ← | — | = | M |] | m | } | | | ユ | ス | ヘ | ン | ○ | |
| | E | S _O | ↑ | . | > | N | ^ | n | ~ | | | ヨ | セ | ホ | ゝ | ◀ | |
| | F | S _I | ↓ | / | ? | O | _ | o | | | | ツ | ソ | マ | ° | ◀ | |

付録I 参考文献&引用文献一覧表

1. 「N-BASIC リファレンス・マニュアル」NEC
2. 「PC-8001 ユーザーズ・マニュアル」NEC
3. 「μCOM-82 ユーザーズ・マニュアル」NEC
4. 「μCOM-82 インストラクション活用表」NEC
5. 「月刊マイコン」1981 1月号～1982 12月号「マシン語講座」／川村 清／電波新聞社
6. 「Z-80 マイコン・プログラムテクニック」電波新聞社
7. 「PC-8001 BASIC SOURCE PROGRAM LISTINGS」
秀和システムトレーディング株式会社
8. 「PC-8001 マシン語活用ハンドブック 初級編」秀和システムトレーディング
株式会社
9. 「PC-8801 解析マニュアル」秀和システムトレーディング株式会社
10. 「月刊アスキー」1979 11月号「2D-4M」／乾 謙一／株式会社アスキー
11. 「DUAD-PC ユーザーズ・マニュアル」株式会社アスキー
12. 「DUAD-88D ユーザーズ・マニュアル」株式会社アスキー

DEMPA マイコン BOOKS

マイコン雑誌
これで
決定!!



月刊
マイコン別冊 **PC-9801活用研究Ⅰ**

ビジネス
ソフト編

PC-9801活用研究
ビジネスソフト編



■日本電気の16ビットパソコンPC-9801活用研究の第1弾。ビジネスソフトの基本である、販売管理、仕入管理、在庫管理、給与計算のプログラムを紹介。付録としてPC-9801ソフトオールカタログ付。
定価 250円

B5判 222頁 定価 1,500円

月刊
マイコン別冊 **PC-9801活用研究Ⅱ**

グラフィック
マスター編

PC-9801活用研究
グラフィックマスター編



■日本電気のベストセラーマイコンPC-9801活用研究の第2弾。本機の特徴であるグラフィック機能を徹底的にマスターするために、カラー写真をつけた楽しいプログラムを満載。PC-9801のユーザーの「福音の書」です。
定価 250円

B5判 240頁 定価 1,500円

OA
情報別冊 **IBM5550解体新書**

オフィス革命児
5550のすべて!

IBM5550
解体新書



■1台三役のキャッチフレーズに発売されたIBMマルチステーション5550の機能・特徴を徹底解剖。オフィス革命児5550のすべて!すぐに使えるプログラムリストは、実用書として威力発揮!
定価 300円

A4変型判 200頁 定価 1,200円

月刊
マイコン別冊 **マイコンBASIC講座①**

実務応用編

マイコンBASIC講座①
実務応用編



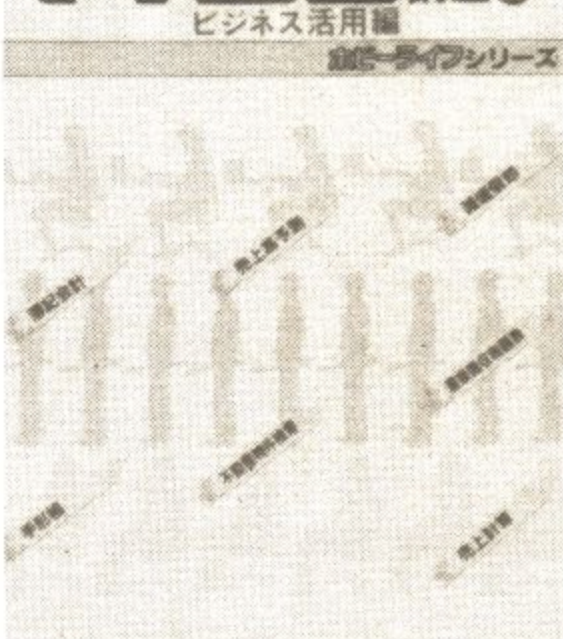
■だれにも理解できる入門的な内容がほとんどで、マイコンがわからないという人に読んでもらい、本当の意味でマイコンを活用できるようにまとめた入門書。
定価 300円

B5判 196頁 定価 1,300円

月刊
マイコン別冊 **マイコンBASIC講座②**

ビジネス
活用編

マイコンBASIC講座②
ビジネス活用編



■解りやすくビジネス利用基本から、活用まで身近な項目を選び、プログラム作成に役立つようにマイコンを初めてタッチする人が問題とする点、弱点について自分自身の経験を通して解説している。
定価 300円

B5判 193頁 定価 1,300円

月刊
マイコン別冊 **マイコンBASIC講座③**

プログラム
マスター編

マイコンBASIC講座③
プログラムマスター編



■言語は機種によって多少の方言があるが、本書は方言が少ない共通語ともいえるBASIC言語を中心にまとめ、パソコンの生いたちからアニメーションテクニックまで入門者向けにわかりやすく解説。
定価 300円

B5判 231頁 定価 1,400円

月刊
マイコン別冊 **マイコンBASIC講座④**

データファイル
活用編

マイコンBASIC講座④
データファイル活用編

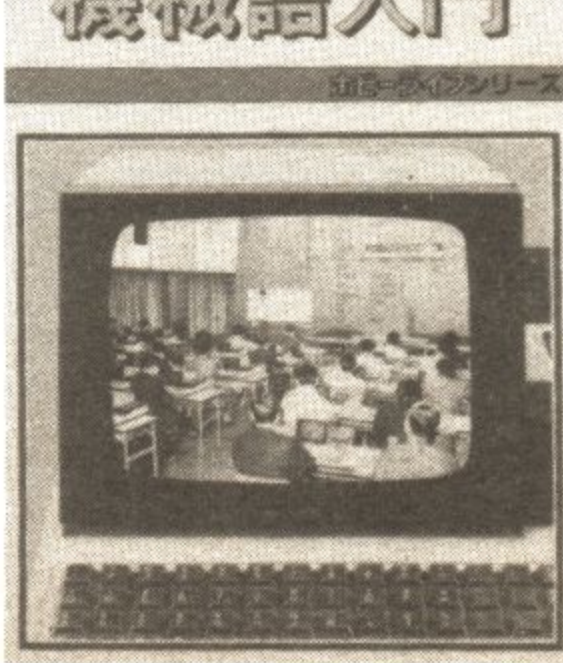


■「データ・ファイル」の処理テクニックに特に重点をおき、実務に欠かせないフロッピーディスク装置を使用する上での、データ入・出力、並べかえ、検索、ファイル管理等のプログラミングノウハウを詳しく紹介。
定価 300円

B5判 252頁 定価 1,400円

月刊
マイコン別冊 **マイコン機械語入門**

マイコン機械語入門



■機械語というのは機種ごとにそれぞれ違い、単に一般論を述べても焦点がぼけてしまう。本書ではシャープのMZ-80K/C、1200に従って解説をすすめ、ゲーム作りを前提にまとめている為、親しみ易い内容になっている。
定価 300円

B5判 202頁 定価 1,300円

月刊
マイコン別冊 **Z-80マイコンプログラムテクニック**

Z80マイコンプログラム



■ザイログ社の開発したCPU Z80が、手軽さと汎用性のため、8ビットマイクロコンピュータの主流になっている。本書はZ80の命令をわかりやすく解説。またテストプログラム、演算プログラムも各種紹介している。
定価 250円

B5判 238頁 定価 1,300円

月刊
マイコン別冊 **PC-8001マシン語入門**
PC-8801

マシン語の基礎から
ゲームの製作まで

■あなたは自分のマシンを使いこなしているか？ 現在BASICを理解できる人は、マイコン所有者の10分の1、マシン語に至っては、そのまた何分の1かである。本書は“マシン語修得のための近道”として作られた。
◎300円



B5判 210頁 定価1,300円

月刊
マイコン別冊 **PC-8001マシン語入門II**

アセンブラから電子音楽付き
カラー・グラフィックまで

■好評のPC-8001、8801マシン語入門に続く第二巻。例題を通して実際に自分の目で確かめながら学習を進めている。今回は、アセンブラから効果音付きカラーグラフィックにまで挑戦。
◎250円



B5判 218頁 定価1,300円

月刊
マイコン別冊 **FM-7活用研究**

ゲームからマシン語・新言語まで

■迷路の中のエサを食べながらオバケの追撃をさけ、逃げるゲーム“パックマン”。ゲームセンターの興奮を本書で再現。他にF-BASICの解説、6809のマシン語の説明等も加わったFM-7の解説書。
◎300円



B5判 244頁 定価1,500円

月刊
マイコン別冊 **PC-8001活用研究**
(ゲーム編)

■PC-8001の機能などを細かく、初心者の方にもわかり易く説明。プログラムも実務的なものから、ゲームまで広範囲にわたって利用でき、またプログラムの選び方、特徴なども紹介されている。
◎250円

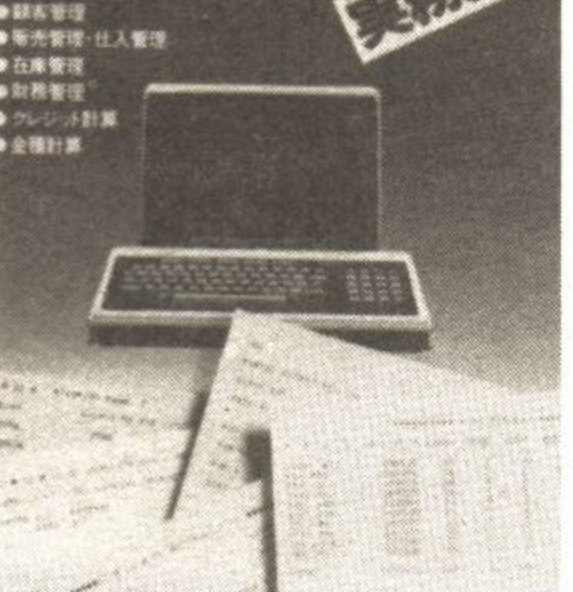


B5判 198頁 定価1,200円

月刊
マイコン別冊 **PC-8001活用研究・実務編**

すぐに役に立つ
ビジネス・ソフト

■NECのPC-8001用ビジネスソフトを6本紹介。プログラムリストとフローチャートの二つを掲載、変数の使い方が難しいものについては変数表をのせた。
◎300円



B5判 154頁 定価1,500円

月刊
マイコン別冊 **GAMINGへの招待**

あなたもテレビゲームプログラムに挑戦

■リアルタイムゲーム、シミュレーションゲーム、思考ゲーム、マイコンパズル等のエキサイティングなマイコンゲームをあなた自身で作ってみたいと思いませんか？ 本書はそのテクニックをあらゆる角度から紹介説明しています。
◎250円



B5判 220頁 定価1,300円

月刊
マイコン別冊 **PC-8001活用研究**
ビジネスグラフィックス

■コンピュータ言語“BASIC”はきわめて修得しやすい言語です。本書はそのBASICを使ってビジネス用プログラムの重要な項目である作表とグラフ作成のプログラムを各種紹介している。
◎300円



B5判 228頁 定価1,500円

月刊
マイコン別冊 **素早くマスタービジネスBASIC**
N5200 モデル05

■パソコンのビジネス活用が進むなかで、NECの最上位16ビットパソコン「N5200モデル05」を使い、イラストを多用し、見ながら自然に理解できるようやさしく解説したビジネスBASICの入門書。
◎250円



B5判 186頁 定価1,500円

月刊
マイコン別冊 **APPLE DOS入門**

ゲームからビジネスまでディスク操作テクニック全公開!!

■APPLE社のAPPLE IIはハード・ソフト共優秀なものを備えて各社のパソコン販売の目標になっています。本書はそのAPPLE IIのディスクを使いこなすためのノウハウをやさしく解説した手引書です。
◎250円



B5判 220頁 定価1,300円

月刊
マイコン別冊 **MZ-2000活用研究**

テクニックから機能・各種アプリケーションまで

■機械の特徴、性能を徹底的に分析、MZ-80Bとの互換性をわかり易く紹介。汎用プログラムとしてゲームプログラム、日本語ワードプロセッサの活用、BASICコンバートプログラム、実務プログラムを各種紹介。
◎250円



B5判 178頁 定価1,300円

月刊
マイコン別冊 **MZ-2000/2200プログラムテクニック**

君のMZを120%使えるマガジンBOX

■これだけはそろえないとユーティリティ、なにかと便利なサブルーチン、集計プログラム、データ通信プログラム等、あなたのMZをTPOに合わせて活用するためのテクニックを各種紹介。
◎300円



B5判 192頁 定価1,300円

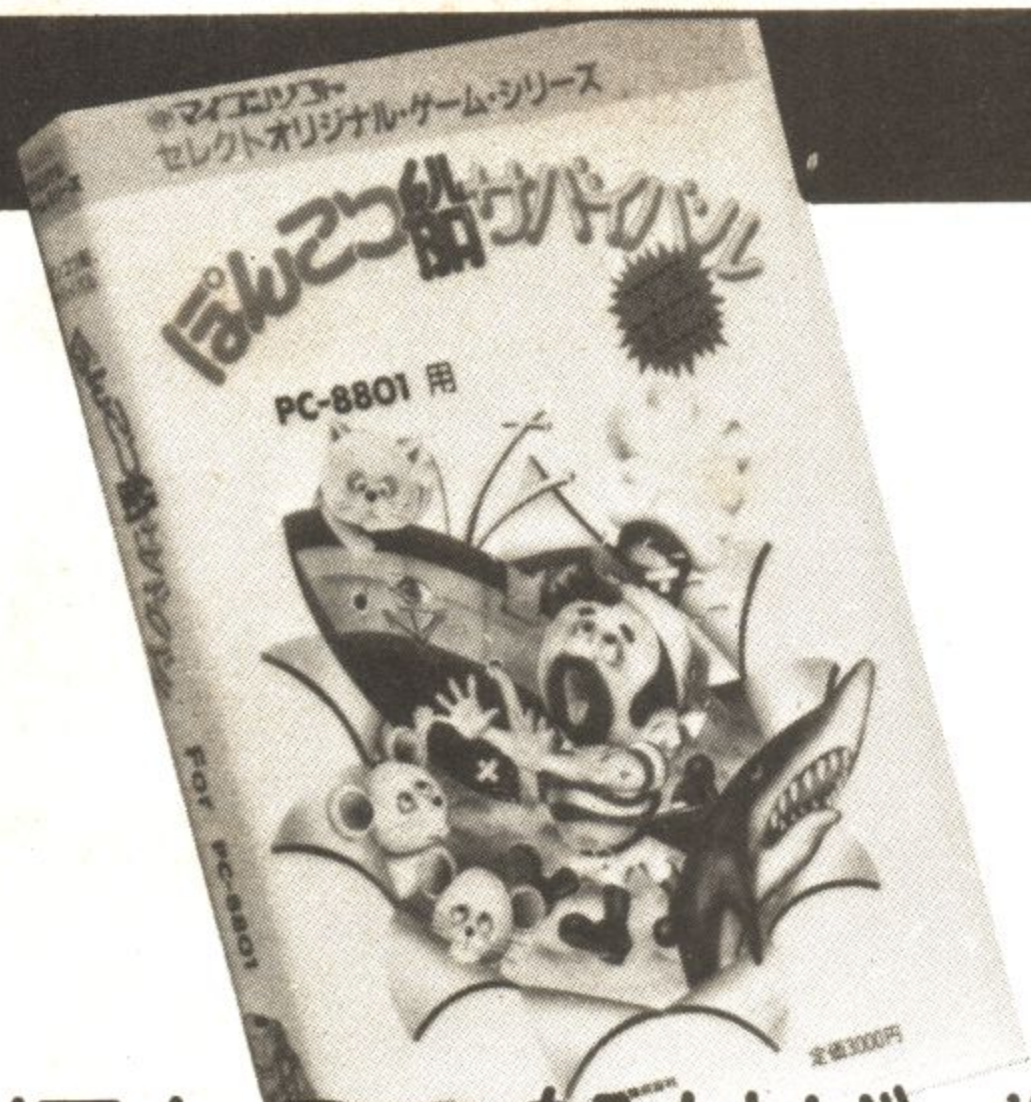
月刊
マイコン別冊 **パソコンワープロ**

今、キミの手にパソコンの秘密兵器が……

■OA時代になくはないのがワープロ。パソコンでもワープロ機能を、という要望に応えたパソコンワープロを本書では詳細に説明。また、ユーザー訪問記及びオールガイドも併録。
◎300円



B5判 168頁 定価1,300円



ぽんこつ船サバイバル

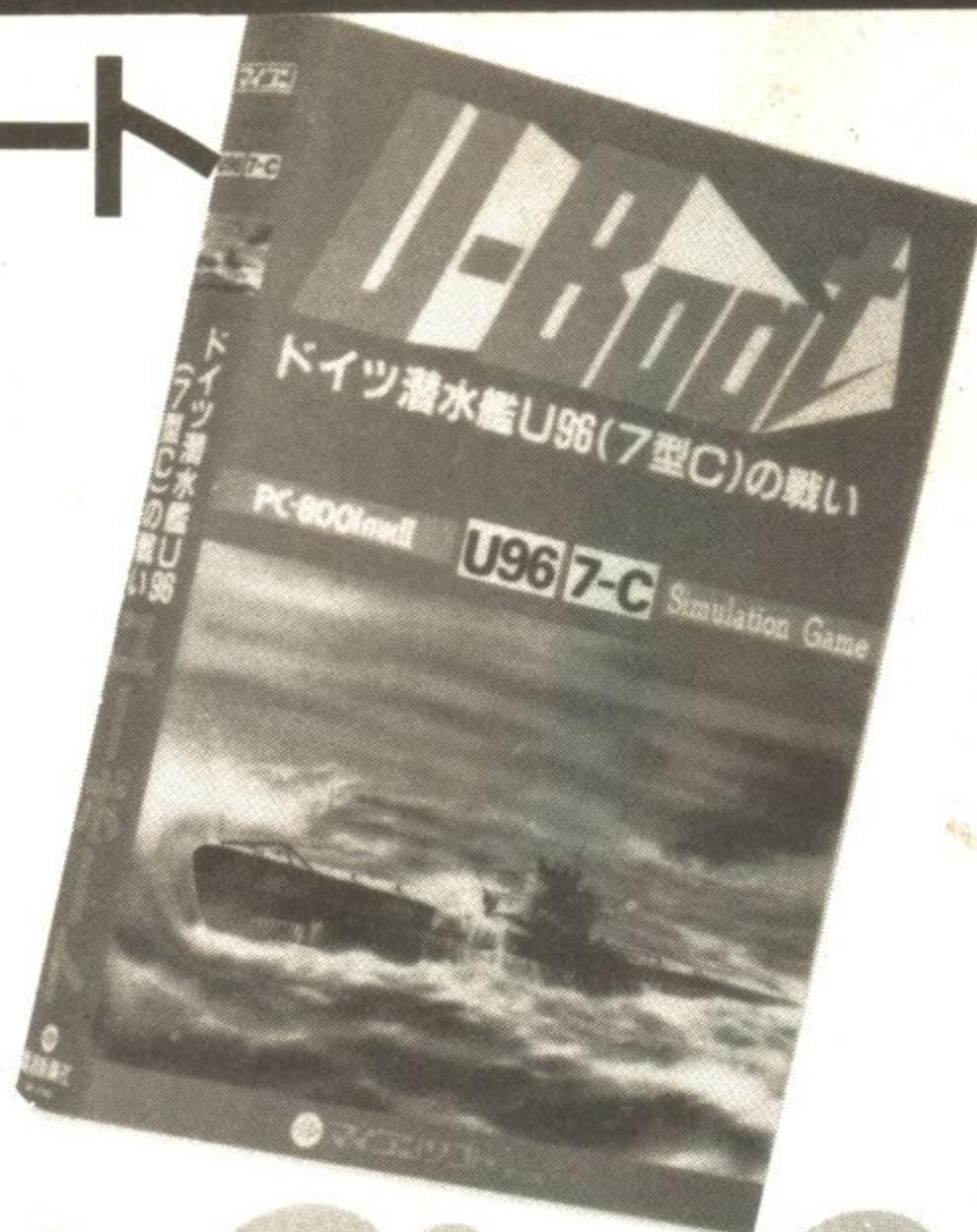
PC-8801 定価 3,000円

ネズミと水夫がくりひろげる底ぬけ反射ゲーム。

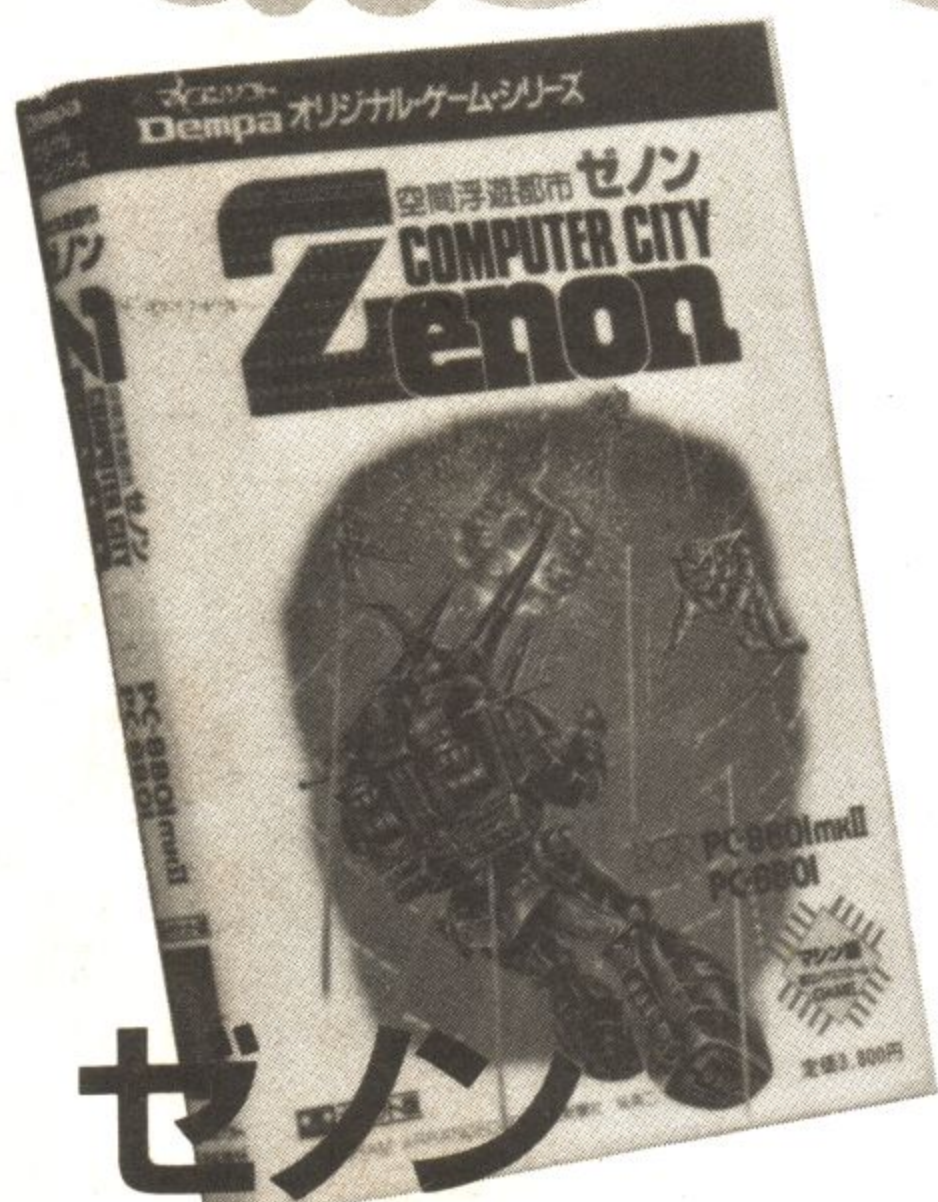
ユーボート

PC-8001mkII
定価 3,800円

作戦指令!!
イギリス輸送船団を
撃沈せよ!



オリジナル・ゲーム・シリーズ



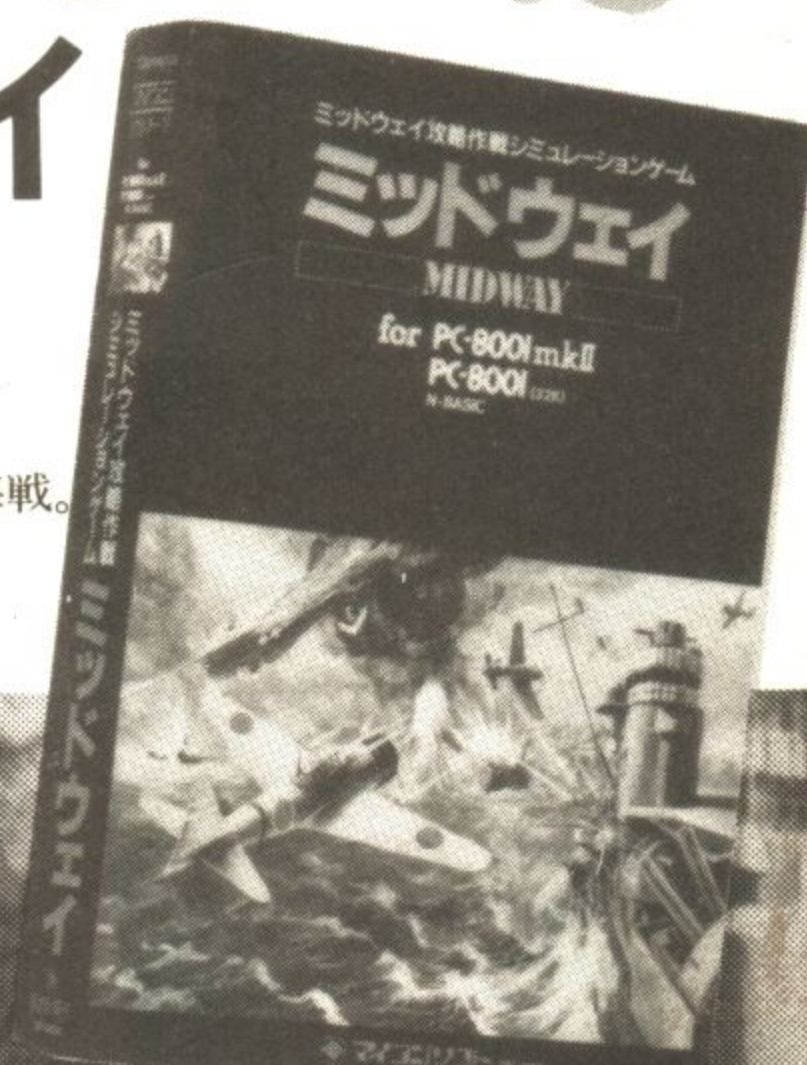
PC-8801/mkII 定価 3,800円

マザーコンピュータにコントロールさ
れた空間浮遊都市ゼノンの市民を守れ!

ミッドウェイ

PC-8001 (N-BASIC)
FM-7・8 (F-BASIC)
定価 3,800円

日本の運命を変えたミッドウェイ海戦。
史実をもとにした日米機動部隊の
シミュレーションゲーム。



マイコン別冊

Z80マシン語プログラム入門

川村 清 著

昭和59年4月3日発行

©1984 Printed in Japan

定価 1,500円 (送料300円)

発行人 平山 秀雄

発行所 (株)電波新聞社

郵便番号141 東京都品川区東五反田1-11-15

電話(03) 445-6111(大代) 振替東京5-51961

印刷所 大日本印刷(株)

製本所 (株)堅省堂

すっきりフロッピー

パソコンと調和、
フロッピーの集約

新登場

M・PCユーザーに
朗報

パソコンと調和、
フロッピーのデザイン



K-305FM (MINI DISK UNIT-DUAL TYPE)

●両面倍密度 ●高性能コントローラー付

¥128,000

ユーザーオプション

●FM-8用 FDC I/Oポート

K-305-11.....¥13,000

●FM-7用 FDD I/Fカード

K-305-12.....¥13,800

●FM-7・8用

接続ケーブル K-305-03.....¥4,000



K-305PC (MINI DISK UNIT-DUAL TYPE)

●両面倍密度 ●インテリジェントタイプコントローラー付

¥128,000

ユーザーオプション

●PC-8001用 FDD I/Oポート

K-305-01.....¥13,000

接続ケーブル/K-305-03.....¥4,000

●PC-8001MK II、PC-8801、PC-9801用

接続ケーブル/K-305-02.....¥5,000

●PC-8011用接続ケーブル/K-305-04.....¥5,000

Logitec

le 関東電子株式会社

マイコン営業部 〒101 東京都千代田区外神田2-15-2新神田ビル ☎03-257-6221
システム営業部 〒101 東京都千代田区外神田2-15-2新神田ビル ☎03-257-6291
大阪支店 〒556 大阪市浪速区日本橋3-3-5カトウビル ☎06-632-0207代

●札幌営業所 ☎011-832-0131 ●仙台営業所 ☎0222-33-0257 ●長岡営業所 ☎0258-32-8888
●宇都宮営業所 ☎0286-34-7505 ●群馬営業所 ☎0270-23-2301 ●多摩営業所 ☎0423-44-8111
●町田営業所 ☎0427-28-8882 ●千葉営業所 ☎0472-48-2955 ●沼津営業所 ☎0559-51-2888
●浜松営業所 ☎0534-64-2238 ●名古屋営業所 ☎052-263-1693 ●京都営業所 ☎075-343-0995
●広島営業所 ☎082-227-5536 ●福岡営業所 ☎092-474-5777 ●熊本営業所 ☎0963-26-1166

マイコン別冊 Z80マシン語プログラム入門 昭和五十九年四月三日発行

たしかに技術で世界をむすぶ

NEC

大人のパソコン新発売。



仕事の達人。遊びの達人。

会社でも、家庭でも、自由に使える高性能パソコンPC-8801mkII、新登場です。使いやすさで定評のある名機、PC-8801のグラフィック機能やワープロ機能、日本語処理機能などをさらに充実。ミニフロッピーディスクドライブも本体に内蔵し、いちだんと便利になりました。大人たちの期待に応える実力派です。もちろん、すぐに使える市販のソフトウェアの数は、数千種類。仕事も遊びも見事にこなすヤングアダルトにふさわしい、洗練されたパソコンです。

NECパーソナルコンピュータ
PC-8800シリーズ
PC-8801mkII

PC-8801mkII model 10…本体標準価格168,000円
PC-8801mkII model 20…本体標準価格225,000円
(ミニFDD1台内蔵)
PC-8801mkII model 30…本体標準価格275,000円
(ミニFDD2台内蔵)
※ディスプレイは別売。model 10、20はオプションとしてミニFDDが本体内に2台まで実装できます。

ソフト数千、高感度パソコン。



NECパーソナルコンピュータ
PC-8000シリーズ
PC-8001mkII
本体標準価格……………123,000円

PC-2000シリーズ/ PC-6000 PC-6001mkII本体/ 新発売 PC-6600シリーズ/ PC-8000 PC-8001mkII本体/ PC-8200シリーズ/ 新発売 PC-8800 PC-8801mkII本体
新発売 PC-100シリーズ/ 新発売 PC-9800 PC-9801E/ PC-9800 PC-9801/ 新発売 PC-9801E/ N5200 モデル05

NECのパソコンファミリー



日本電気グループ NECパソコンインフォメーションセンター
〒108 東京都港区三田三丁目14-10(明治生命三田ビル) ☎(03)452-8000(代)



コンピュータ アート コミュニケーション